National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

NL-339 (r. 88/04) c

Canada

University of Alberta

ANNIE: An Adaptive Neural Network for Image Enhancement

by

Dan Perl

A thesis
submitted to the Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree
of Master of Science

Department of Computing Science

Edmonton, Alberta
Spring 1992

National Library of Canada
Bibliothèque nationale du Canada

Canadian Theses Service
Service des thèses canadiennes

Ottawa, Canada
K1A 0N4

The author has granted an irrevocable non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of his/her thesis by any means and in any form or format, making this thesis available to interested persons.

L'auteur a accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

The author retains ownership of the copyright in his/her thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without his/her permission.

L'auteur conserve la propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-73096-X

Canada

UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR: Dan Perl

TITLE OF THESIS: ANNIE: An Adaptive Neural Network for Image Enhancement

DEGREE: Master of Science

YEAR THIS DEGREE GRANTED: 1992

Permission is hereby granted to the University of Alberta Library to reproduce single copies of this thesis and to lend or sell such copies for private, scholarly or scientific research purposes only.

The author reserves all other publication and other rights in association with the copyright in the thesis, and except as hereinbefore provided neither the thesis nor any other substantial portion thereof may be printed or otherwise reproduced in any material form whatever without the author's prior written permission.

(Signed) Dan Perl . . . . . .

Permanent Address:
276 Princess Ave.
Willowdale, Ontario
Canada M2N 3S4

Date: April 1, 1992

We are confronted with insurmountable opportunities.
– Walt Kelly, "Pogo"

UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance, a thesis entitled **ANNIE: An Adaptive Neural Network for Image Enhancement** submitted by **Dan Perl** in partial fulfillment of the requirements for the degree of Master of Science.

. . . . . .
T. A. Marsland (Supervisor)

. . . . . .
W. W. Armstrong

. . . . . .
X. Li

. . . . . .
M. R. W. Dawson (Psychology)

Date: . . . . . .

Abstract

ANNIE is a neural network that removes noise and sharpens edges in digital images. For noise removal, **ANNIE** makes a weighted average of the values of the pixels over a certain neighborhood. For edge sharpening, **ANNIE** detects edges and applies a correction around them. Although averaging is a simple operation and needs only a two-layer network, detecting edges is more complex and demands several hidden layers. Based on Marr's theory of natural vision, the edge detection method uses zero-crossings in the image filtered by the $\nabla^2 G$ operator (where ∇^2 is the Laplacian operator and G stands for a two-dimensional Gaussian distribution), and uses two channels with different spatial frequencies. Edge detectors are tuned for vertical and horizontal orientations. Lateral inhibition implemented through one-step recursion achieves both edge relaxation and correlation of the two channels. Training by means of the quickprop algorithm determines the shapes of the weighted averaging filter and the edge correction filters, and the rules for edge relaxation and channel interaction. **ANNIE** uses pairs of pictures as training patterns: one picture is a reference for the output of the network and the same picture deteriorated by noise and/or blur is the input of the network.

Contents

1 Introduction 1

1.1 About Image Enhancement 2

1.2 About Natural Vision 3

1.3 About Neural Networks 5

2 Theoretical Background of Edge Detection 8

2.1 Center-Surround Filters and Zero-Crossings 9

2.2 Spatial-Frequency Channels 11

2.3 Edge Relaxation. Its Relation with Lateral Inhibition in Neural Mechanisms 13

2.4 Related Work 14

3 Implementation 16

3.1 The Method Used to Enhance Images 16

3.2 The Operation of ANNIE 20

3.2.1 The Operation of the Output Layer 25

3.2.2 The Operation of the Edge Detection Channels 26

4 The Training Method 36

4.1 Overview of Backpropagation 37

4.2 Quickprop as an Improvement to Backpropagation 38

4.3 Error Backpropagation and Weight Changes for ANNIE 40

4.4 Constraints in Updating Weights 42

5 Simulations 44

5.1 Methodology of Simulation 47

5.2 Simulation Results 52

5.3 Comparison of Results 57

6 Conclusions 59

A Edge Sharpening in the Presence of Noise 68

B Noise Removal 74

C Edge Sharpening 84

D Using ANNIE on Real Pictures 88

List of Figures

1	The shape of the Mexican-hat operator.	10
2	The response of the Mexican-hat operator to an isolated step edge.	11
3	The general structure of **ANNIE**.	17
4	The shape of the averaging filter.	18
5	The shape of an edge correction filter.	18
6	Sample input pattern.	21
7	The architecture of **ANNIE**.	22
8	The links entering the output layer.	25
9	The output for the sample picture.	26
10	The sample picture processed by Mexican-hat layers.	28
11	The array of outputs of the Mexican-hat layers, used for zero-crossing detection.	29
12	Zero-crossings detected in both channels for the sample picture.	31
13	The links inside and between the two channels.	34

14 Edges identified in both channels for the sample picture. . . . 35

15 The pictures used for training. 45

16 The picture used for testing. 47

17 Degradations applied to one of the training patterns. 50

18 The output in response to the test pattern, compared to the input and to the reference, after training for noise removal and edge sharpening. 54

19 Weights obtained by training for noise removal and edge sharpening. 55

20 The output in response to the test pattern, compared to the input and to the reference, after training for edge sharpening in the presence of noise. 71

21 Weights obtained by training for edge sharpening in the presence of noise. 72

22 The output in response to the test pattern, compared to the input and to the reference, after training for noise removal with noise generated randomly in the range [-0.15, 0.15]. 76

23 Weights obtained by training for removing noise generated randomly in the range [-0.15, 0.15]. 77

24 The output in response to the test pattern, compared to the input and to the reference, after training for noise removal with noise generated randomly in the range [-0.3, 0.3]. 80

25 Weights obtained by training for removing noise generated randomly in the range [-0.3, 0.3]. 81

26 The output in response to the test pattern, compared to the input and to the reference, after training for edge sharpening. 85

27 Weights obtained by training for edge sharpening. 86

28 Results obtained on real pictures. 89

29 Comparison between **ANNIE** and median filtering. 91

List of Tables

3.1	The links between **ANNIE**'s layers.	24
5.2	The mean square error for noise removal and edge sharpening.	53
5.3	The summary of the results for the test pattern.	57
A.4	The mean square error for edge sharpening in the presence of noise (simulation with an experimental, larger network structure).	69
A.5	The mean square error for edge sharpening in the presence of noise (simulation with a reduced network structure).	70
A.6	The mean square error for edge sharpening in the presence of noise.	70
B.7	The mean square error for noise removal with noise generated randomly in the range [-0.15, 0.15].	75
B.8	The mean square error for noise removal with noise generated randomly in the range [-0.3, 0.3].	75
C.9	The mean square error for edge sharpening.	84

Chapter 1

Introduction

The purpose of this work is a practical one: to perform noise removal and edge sharpening, two operations of image enhancement. The method is based on an adaptive neural network. The algorithms traditionally used in image enhancement are rarely related to techniques from natural vision. However, because nature still surpasses these artificial methods, emulating some of nature's techniques is not only challenging from a scientific point of view, but also practical. Moreover, such emulations are more feasible today, thanks to the emergence of artificial neural networks. **ANNIE** is an attempt in this direction. It combines natural vision techniques with artificial neural networks for the purpose of image enhancement.

Before presenting the implementation of **ANNIE**, it is necessary to introduce the three scientific areas to which this work is related. The first section lays down the purpose of digital image enhancement and some traditional techniques of noise removal and edge sharpening. The second section offers a background in the neurophysiology of natural vision; the emphasis is put on the processes that are related to noise removal and edge detection. Finally, the third section gives an outline of artificial neural networks and their applications.

1.1 About Image Enhancement

Whenever a picture is converted from one form to another, for instance digitized, copied, scanned, transmitted, or displayed, the quality of the resultant output may be lower than that of the input. Among such image degradations, blur and noise are two of the most frequent. Some effects of the degradation processes can be compensated by image enhancement techniques [1, 2]. Because blurring is an averaging, or integration operation, most sharpening techniques are based on differentiation operations, particularly on the Laplacian operator, $(\partial^2/\partial x^2 + \partial^2/\partial y^2)$. On the other hand, because blurring weakens high spatial frequencies more than low ones, some sharpening techniques use filtering to emphasize the high spatial frequencies of pictures. However, both differentiation and high-frequency filtering cannot be used indiscriminately to sharpen a picture when it is noisy as well as blurred. These methods have to be restricted to frequency ranges where the picture is stronger than the noise, because noise generally involves high rates of change in gray level and it usually becomes stronger than the picture signal at high frequencies. When knowledge about the blur that is to be corrected is available, restoration techniques such as inverse filtering are used instead.

In noise removal and smoothing techniques, the basic difficulty is that they blur the picture when applied indiscriminately. There are methods that permit smoothing without introducing undesirable blurring, but they are more complex than simple averaging and can be used only in restricted cases. To smooth a picture without blurring, averaging can be used if it is performed only at selected points or only with selected neighbors, in such a way that averaging never happens across edges. For this purpose, an edge (or line) detection operator has to be applied, and then averaging has to be performed only at points where such features are not present. When noise consists of isolated points that contrast with their neighbors (e.g., "salt-and-pepper" noise), noise removal is easy. Such a noise can be detected by comparing each point's gray level to the levels of its neighbors. If the point is substantially different from all its neighbors (or nearly all of them), it can be

classified as a noise point, and it can be removed by interpolation, replacing it with the average of the neighbors. A powerful smoothing technique that does not blur edges is median filtering, in which a value at a point is replaced by the median of the values in a neighborhood of the point. Rather than the median, the value to be taken by the point can be the lowest, highest, or middle ranking value — even any desired rank — like the mode of the neighborhood (i.e., the value that occurs most often). However, this technique can destroy thin lines and isolated points, and it can clip corners.

All the image enhancement techniques presented above are heuristic in nature. They have several parameters that must be adjusted to suit the characteristics of the degradations to be removed. These parameters must be tuned by trial-and-error procedures [1]. For example, edge detection [3] can be performed by applying a threshold operation, preferably after enhancing the gray scale of the picture. But a threshold level set too high will not permit detection of low-amplitude structural image elements, and conversely, a threshold level set too low will cause noise to be falsely detected as an image edge. Because the resulting performance can be only measured, not predicted, it is difficult to determine a direction for optimization.

1.2 About Natural Vision

The human visual system has a complexity and a degree of excellence not even remotely equaled by any of the artificial systems. However, natural vision itself is not perfect, because it can create visual illusions like neon color spreading, Mach bands, brightness and color illusions, and others [4]. This is why **ANNIE** uses only some neural techniques for image processing but does not try to replicate the exact neural architecture used by the brain. The intention is only to use some mechanisms of the visual system as a basis for edge detection. This section presents information about these functions in human vision to support the neural network architecture used by **ANNIE**.

[1] In an adaptive neural network, such parameters are tuned by training.

The retina is the part of the vision system that is studied most and understood best [5, 6]. The vertebrate retina consists of five classes of neurons: receptor, bipolar, horizontal, amacrine, and ganglion cells. Receptors are the first stage in the information path. Ganglions are the last stage and represent the only connection that the retina has with the cortex. There are two types of ganglion cells, each of which responds differently to illumination patterns. Both types do not convey information about absolute levels of illumination. The depolarizing ganglions are activated in response to central illumination of their receptive field and are inhibited in response to surround illumination. The second type, the hyperpolarizing ganglions, are inhibited in direct illumination and are activated in surround illumination. This mechanism makes ganglion cells sensitive to edges crossing between the opposing regions of the receptive field.

The retina is characterized by having several receptive-field sizes [7, 8]. This has generated the idea of channels sensitive to particular bands of spatial frequencies. Their existence can be proved psychophysically in man by adaptation experiments [9]. The first evidence for these channels was found by Campbell and Robson [10]. They stated that "a picture emerges of functionally separate mechanisms in the visual nervous system, each responding maximally at some particular spatial frequency and hardly at all at spatial frequencies differing by a factor of two[2]."

From the retina, signals proceed via the optic nerve to the lateral geniculate. Here, no further processing but only a re-encoding of the information is performed. Then the signals proceed to the primary visual cortex. Its cells are divided into four categories: simple, complex, hypercomplex, and higher order hypercomplex [11]. Just like ganglion cells in the retina, cortical cells are not significantly influenced by diffuse illumination of the retina. There are several types of simple type cells, detecting special features like edges or narrow bars, oriented at specific angles. The width of the narrow light or dark bar is comparable to the various diameters of the center regions in the receptive field of ganglion or lateral geniculate cells. The

[2]The importance of separate mechanisms with different spatial frequencies is discussed in Chapter 2.

spot-like contrast representation of ganglion cells is just transformed and extended into a line or an edge.

Complex cells also require a specific field axis orientation of a dark-light boundary, while illumination of the entire field is ineffective. The demand, however, for precise positioning of the stimulus is relaxed. In addition, there are no longer distinct on- and off- areas. Complex cells signal the abstract concept of orientation without strict reference to position. In hypercomplex cells, the best stimulus still requires a certain orientation, but also involves some discontinuity, such as a line that stops, an angle, or a corner. And again, as in simple cells, position within the receptive field is important.

1.3 About Neural Networks

A recent interest in artificial neural networks has been brought during the last decade by recent advances in technology and a better understanding of how the brain works. Consequently, neurocomputing is establishing itself more and more as an alternative form of information processing. Neural networks have already proven themselves to be good in many applications for which conventional computers are bad. They do well at solving complex pattern-recognition problems such as understanding continuous speech, identifying handwritten characters, or at performing operations like optimizations and market or weather forecast. Signal processing and image processing have been two of the areas most explored for neural networks applications.

Nearly all automated information processing is at present based on an algorithmic approach. To execute it, a function must be understood and an algorithm has to be implemented for it. But there are complex tasks for which it is virtually impossible to devise a series of logical or arithmetical steps that will arrive at the answer. However, in some of these cases it is possible to specify the tasks exactly and even develop an endless set of examples of the function being carried out. More so, they are implementable, since humans are capable of doing them. A general

characteristic of these tasks is that they involve associating objects in one set with objects in another set. Reading aloud a text is an example of associating groups of letters, spaces, and punctuation with specific sounds, pauses, and intonations.

Another motivation for the recent interest in neurocomputing is its massively parallel nature. As new and more powerful parallel computers are developed, neural networks stand as one of their most natural applications. Straightforwardly implementable on parallel architectures [3], and with a great power of adapting, neural networks offer the fault-tolerance and speed of natural brain.

There are several introductions to neural networks [12, 13]. A neural network consists of a collection of processing elements. Each processing element has many input signals, but only a single output signal. The output signal fans out along many pathways to provide input signals to other processing elements. These pathways connect the processing elements into a network. The processing that each element does is determined by a transfer function — a mathematical formula that defines the element's output signal as a function of the input signals. Often a neural network is divided into layers — groups of processing elements that receive inputs from the same other layers and have the same transfer function.

In general, every connection entering a processing element has an adaptive coefficient called a weight assigned to it. These weights determine the strength of the connections from other processing elements. Moreover, they are not fixed but may change. Most transfer functions include a learning law — an equation that modifies all or some of the weights according to the input signals and the values resulting from applying the transfer function. It is with these changes that a neural network adapts itself and therefore learns.

Several learning paradigms exist. Detailed classifications of learning methods have been presented [14, 15]. Broadly, learning procedures can be divided into three classes: supervised procedures requiring the specification of the desired out-

[3] **ANNIE** is not implemented in a parallel manner, because an appropriate parallel computer (with fine grain parallelism) was not available. However, it can be modified for such an implementation.

puts, reinforcement procedures requiring the specification of a quality measure for the outputs, and unsupervised procedures that build internal models that capture regularities in their inputs, without receiving any additional information. Among supervised learning paradigms, variations based on backpropagation [16] have emerged as being the most widely used.

Two separate schools have evolved in modeling neural networks, focusing on different goals. In biological modeling the goal is to study the structure and function of real brains in order to explain biological data on aspects such as behavior. In technological modeling the goal is to study brains in order to extract concepts to be used in new computational methodologies. To achieve this objective, it is even admissible to incorporate features in a model even if those features are not neurobiologically possible. **ANNIE** falls into the frame of this latter school.

Chapter 2

Theoretical Background of Edge Detection

The task of low-level vision is to map from the original pattern of light intensities to intermediate abstract representations, providing the positions of surfaces in the environment, their depths, and how they are moving. High-level vision processes receive these representations and use knowledge of the world to postulate what objects there could be that have surfaces positioned, shaped, or moving in the observed ways.

One of low-level vision's tasks is segmentation, the process of extracting information about areas of the image (called regions or segments) that are visually distinct from one another and are continuous in some feature. This feature is usually intensity, but possibly color or less often some textural feature. Segmentation can use edge detection as a preliminary stage, where edges represent local points of discontinuity in that particular feature. In the case of intensity discontinuities, some edge models include ramps along with steps in the definition of edges. We will consider only steps or approximate steps (i.e. steep ramps) [1]. Edges can be described by their image location (usually related to a pixel), their direction (usually aligned with

[1] Because we are concerned with edges as they are perceived in natural vision, it is worth mentioning that Shapley and Tolhurst [17] show that broad ramps and steps are detected by different mechanisms of the visual system.

the direction of maximum change in the feature being used), and their magnitude (which measures the amount of change in the feature).

Several vision theories have emerged in recent years. In a widely accepted vision theory, Marr [18] builds a "primal sketch" (low-level description of an image) from information that is gathered through edge detection. The edge detection theory that Marr uses for this purpose was developed by himself and Hildreth [19] and is at the basis of **ANNIE**'s implementation.

2.1 Center-Surround Filters and Zero-Crossings

In their theory, Marr and Hildreth choose the operator $\nabla^2 G$ for edge detection, where ∇^2 is the Laplacian operator $(\partial^2/\partial x^2 + \partial^2/\partial y^2)$ and G stands for the two-dimensional Gaussian distribution

$$G(x, y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2+y^2}{2\sigma^2}},$$

which has standard deviation σ. $\nabla^2 G$ is then a circularly symmetric Mexican-hat shaped operator whose distribution in two dimensions may be expressed in terms of the radial distance r from the origin by the formula

$$\nabla^2 G(r) = \frac{-1}{\sigma^3\sqrt{2\pi}} \left(2 - \frac{r^2}{\sigma^2}\right) e^{\frac{-r^2}{2\sigma^2}}. \tag{2.1}$$

The shape of the Mexican-hat operator is shown in Figure 1. There are two reasons behind the choice of this operator:

1. It is a differential operator, taking the second spatial derivative of the image. Edge detection usually uses differential operators because a sudden intensity change will cause a peak or a trough in the first derivative or, equivalently, will cause a *zero-crossing* in the second derivative. The Laplacian is also the lowest-order isotropic differential operator. Its orientation-independence makes it more computationally efficient.

2. It is capable of being tuned to any desired scale. This is necessary because intensity changes occur at different scales in an image, and so their optimal

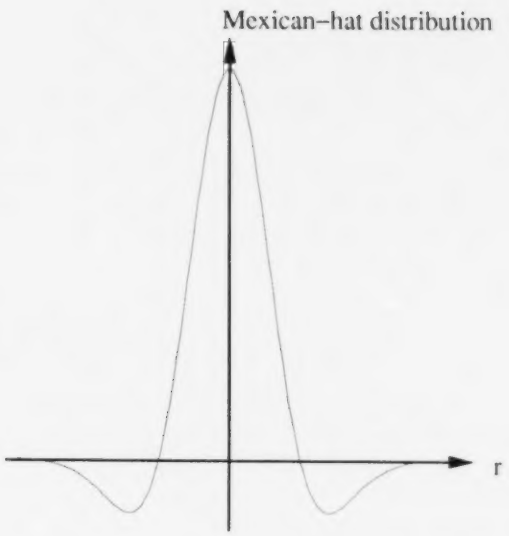


Figure 1: The shape of the Mexican-hat operator.

detection requires the use of operators of different sizes. Sharp edges are detected at all scales. Large filters can detect soft or blurred edges and overall illumination changes, while small filters cannot do that. Also, large filters can smooth out important discontinuities, while with small filters, noise in the picture may introduce edge irregularities. On the other hand, large filters have a lower accuracy in detecting edges that are close to each other; small filters can detect finer details.

The response to this operator for an isolated edge is shown in Figure 2. The shape of this response suggests that the way to detect intensity changes at a given scale is to filter the image with the $\nabla^2 G$ operator and to locate the zero-crossings in the filtered image. It is not necessary to use the zero values for detecting zero-crossings. Instead, it is possible to take advantage of the peak positive value and the peak negative value lying on each side of the zero-crossing. Marr and others [20] prove that the $\nabla^2 G$ transformation incurs no loss of information, meaning that zero-crossing maps obtained at different scales can represent the original image completely.

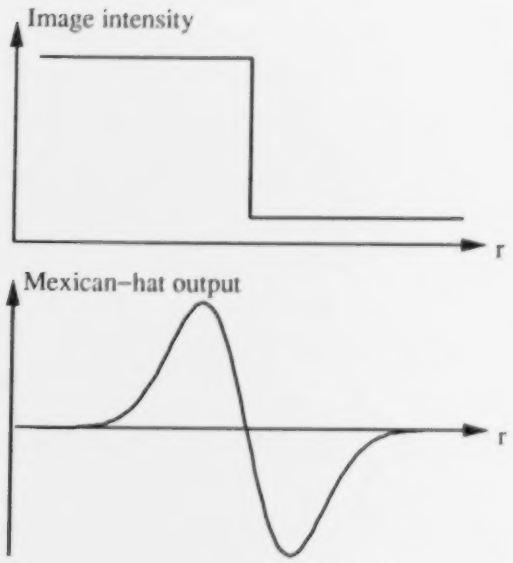


Figure 2: The response of the Mexican-hat operator to an isolated step edge.

A parallel can be drawn between this mechanism and natural vision. It makes the $\nabla^2 G$ equivalent to the on-center and off-center ganglions in the retina, while zero-crossing detection would be achieved by simple cells in the visual cortex[2].

2.2 Spatial-Frequency Channels

As the previous section shows, Marr [18, 19] suggests the use of filter channels of different sizes for detecting intensity changes at different scales. Related work supports this idea: the efficiency of using multiple resolutions has been stressed in image processing [21, 22, 23], and experiments [10, 9] have proven the existence of spatial-frequency channels in the visual cortex. Carpenter together with Grossberg and Mehanian [24] also use multiple space scales in boundary detection to achieve a trade-off between boundary localization, noise suppression, and boundary completion.

[2]See Section 1.2 for a discussion of natural vision.

If a zero-crossing is present in an image filtered through $\nabla^2 G$ at one size, then it should be present at the same location for all larger sizes. It can cease to be so, at some larger size, for one of two reasons: either two or more intensity changes that are close to each other interfere in the larger channel, or intensity changes of different scales are superimposed in the same region. This means that in the general case, the zero-crossings from independent channels of adjacent sizes coincide and can be taken together; they may not coincide in two special cases. In the first case this occurs because of edges that are close to one another and so can be accounted by the outputs of the smaller channel; the false edges outputted by the larger channel must be discounted. In the second case, the zero-crossings probably arise from distinct surfaces or physical phenomena and some edges are accounted by the large channels, some by the small channels. It follows that if there is a range of channel sizes, reasonably well separated in the frequency domain and covering an adequate range of the frequency spectrum, rules can be derived for combining their zero-crossings into a description whose primitives are physically meaningful. The channels must interact to correlate the zero-crossings and thus identify the real edges.

The rules that Marr suggests for channel interaction are complex because several special cases must be taken into account. Because edges that are close to one another can confuse larger channels, they have to be detected explicitly in the smaller channels as thin bars, blobs or terminations. More new descriptive elements are necessary for the cases in which larger channels are recording different physical phenomena. As Section 1.2 shows, the visual cortex contains such detectors, and their existence supports the vision system proposed by Marr. The task of detecting features like thin bars, blobs, and edge terminations is computationally too complex and is avoided in **ANNIE**'s implementation. Instead of channel interaction as proposed by Marr, **ANNIE** uses lateral inhibition between the units in different channels and in the same channel. This alternative cannot offer the same results in preventing the detection of false edges, but it is more easily implementable. Besides that, lateral inhibition is a natural characteristic of neural systems and, as the next

section shows, it can improve feature detection. The experiments show that with this method, **ANNIE** achieves the identification of the correct edges.

Research in human vision [25] points out that when a stimulus activates two or more mechanisms, detection of the stimulus can use the response of either mechanism, and so detection should combine probabilistically all the activated psychometric functions. **ANNIE** follows this principle and detects edges with a weighted sum of the outputs from two channels. The weights involved are determined through training.

2.3 Edge Relaxation. Its Relation with Lateral Inhibition in Neural Mechanisms

One way to improve edge detection measurements is to adjust them based on measurements of neighboring edges. The existence of a weak edge should be reinforced if its direction is compatible with neighboring edges and it should be inhibited in the contrary case. The process of propagating local constraints is analogous to the relaxation methods used in numerical analysis to solve linear equations. Based on this process, relaxation algorithms can be developed to provide an efficient low-level process for machine vision [26, 27, 28].

Relaxation has an obvious equivalent at the level of neuron interactions in the brain: lateral inhibition is a pervasive feature of the wiring of arrays of neurons and it has been studied in both visual systems and somato-sensory systems. It manifests itself through a peak of excitatory connections between nearby cells and a trough of inhibition between cells somewhat further apart. Lateral inhibition at the receptors level has itself an effect on sharpening edges. This process is not used in the implementation of **ANNIE**, because it can introduce unwanted effects (like Mach bands) [4]. Still, the possibility that it could bring an improvement is not excluded. More research is necessary for a confirmation.

2.4 Related Work

Grossberg and Mingolla [29] propose a theory of form perception that can explain some illusory contour formations. According to this theory, two types of perceptual process — boundary completion and featural filling-in — work together to synthesize a final percept. The two distinct types of computations are carried out within parallel systems during brightness, color, and form perception. These two systems are called the *boundary contour system* (BCS) and the *feature contour system* (FCS).

Boundary contour signals are used to generate perceptual edges, both real and illusory. Building up boundary contours is started by the activation of oriented masks at each position of perceptual space. These oriented masks are sensitive to the edges' orientation and to their amount of contrast, but not to the edges' direction of contrast. The outputs from the masks activate three successive stages of interaction. The first stage implements the competition between like orientations at different (but nearby) positions, and is followed by a stage of competition between perpendicular orientations at the same position. The outputs from this second stage input to a cooperative process that achieves boundary completion.

More recently, Carpenter et al. [24] have implemented a circuit for boundary segmentation, called CORT-X filter, that detects, regularizes, and completes image edges in up to 50% noise, while simultaneously suppressing the noise. The processing levels of the CORT-X filter are analogous to those of the Grossberg-Mingolla Boundary Contour System, but contain only feed-forward operations that are easier to implement in hardware. CORT-X also uses two interacting spatial scales to resolve a design trade-off that exists between the properties of boundary localization, boundary completion, and noise suppression.

In both works described above, the authors support their theory with neural data. The network nodes are analogous to cortical simple cells, complex cells, hypercomplex cells, and unoriented and oriented cooperative cells, while the interactions of competition and cooperation between nodes are similar with the process of lateral inhibition existent in the brain. The BCS and the CORT-X are concerned only with

the logical detection of edges and neglect the aspects of edge intensity of contrast and direction of contrast. The networks cannot learn, and therefore cannot adapt themselves [3]. They use many parameters that have to be tuned heuristically; they also use many constraints in the interaction between nodes, based on assumptions that are hard to be verified.

Edge detection from zero-crossings has been implemented in hardware by a Caltech team [30]. Their one-dimensional 64 pixel, analog CMOS VLSI chip localizes intensity edges in real-time, using on-chip photoreceptors. To approximate the Laplacian of the Gaussian $\nabla^2 G$, they replace the difference of gaussians (DOG) proposed by Marr and Hildreth [19] with a difference of exponentials (DOE), more easily implementable in silicon. The chip outputs the logical presence or absence of zero-crossings and their slopes as a measure of the edge's contrast intensity. To suppress the noise inherent in the silicon implementation of photoreceptors, only zero-crossings having the slope over a certain threshold are detected. A two-dimensional version of the chip was only simulated.

[3]Grossberg, one of the most prominent figures in the field of neural networks, opposes the backpropagation learning algorithm for not having a neurophysiologigical equivalent. He is the main exponent of a school of thought claiming that neural networks should be based only on replicating the natural brain.

Chapter 3

Implementation

ANNIE is implemented in the C programming language, and uses the Rochester Connectionist Simulator (RCS) [31], which is a package of programs developed at the University of Rochester. Because RCS is inefficient in terms of speed and use of memory, only some of its features were used. Replacing some of the RCS features causes **ANNIE** to have a structure atypical for neural networks. It is not implemented as an architecture of independent units connected with each other and capable of being simulated on different processors. Instead, its units and the links between them are only simulated by numerical computation. In terms of structure and behavior, however, **ANNIE** is equivalent to a neural network, and therefore, its architecture will be described in a neural network framework.

3.1 The Method Used to Enhance Images

Figure 3 illustrates the general structure of **ANNIE**, which performs two enhancement operations on a degraded picture: noise removal and edge sharpening [1]. The output layer is connected to the input layer and to edge detectors grouped into two channels. The information conveyed by the input layer is used for noise removal

[1] Even if there is no blur in the input picture, averaging for noise removal introduces a blur in the output. The edge sharpening operation corrects both the input blur and the blur resulted from averaging.

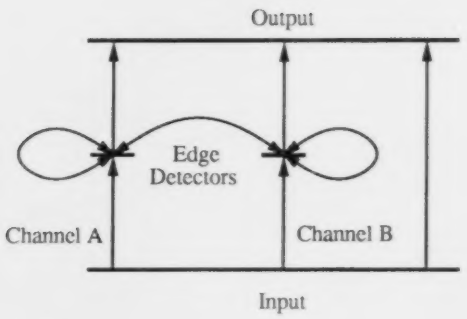


Figure 3: The general structure of **ANNIE**. The arrows indicate communication between units.

through weighted averaging, and the information conveyed by the edge detectors is used for blur correction in the vicinity of edges. The edge detectors provide information about the edges' orientation, direction of contrast, and intensity of contrast. Two orientations are detected: vertical and horizontal. Inclined edges are detected by both detectors and interpolation of the detectors' outputs determines the intensity of contrast. The sign of the detector's output determines the direction of contrast: one direction causes positive outputs, the other direction causes negative outputs. In both cases, the intensity of the edge is proportional to the absolute value of the detector's output.

ANNIE behaves like five filters — an averaging filter and four edge correcting filters (each one of the two channels has one filter for each of the two edge orientations). These filters act separately and their effects are added for giving the final output. The shapes of the filters are determined by the weights of the connections coming into the output unit. Figures 4 and 5 illustrate examples of these shapes, as they resulted from one of the training cases presented in Chapter 5 and in the Appendices. Training the neural network determines the weights of its connections, and therefore the shapes of the five filters. It ensures an optimal combination between averaging and correcting the edges and an optimal combination between the effects of the two edge detection channels. It determines also the edge relaxation for a more accurate detection of edges.

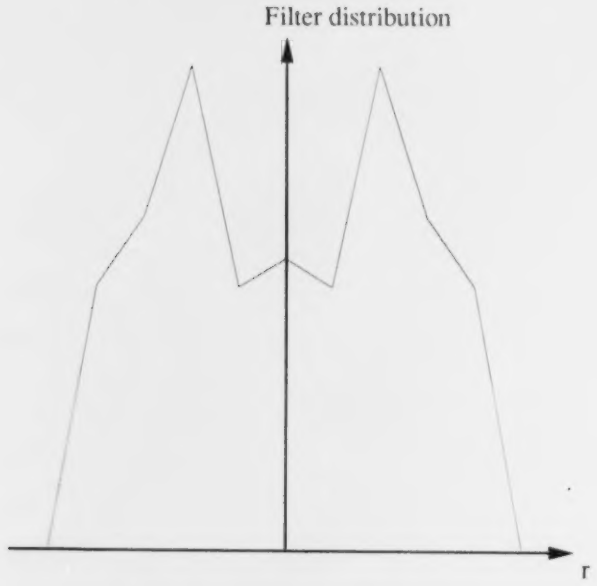


Figure 4: The shape of the averaging filter.

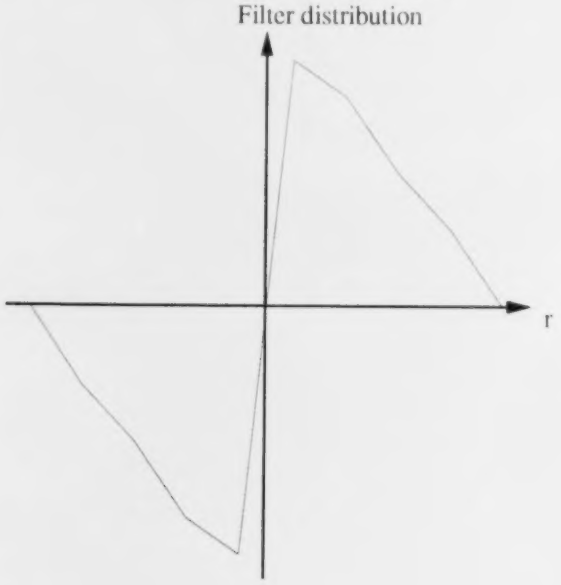


Figure 5: The shape of an edge correction filter.

For an input pattern $X \in R^{n \times n}$, the output $O(X) \in R^{m \times m}$ can be regarded as the sum of the outputs from the two channels (two filters each) and from the averaging filter:

$$O(X) = A(X) + B(X) + W(X), \tag{3.1}$$

where $A(X)$ is the output of channel A, $B(X)$ is the output of channel B, and $W(X)$ is the output of the averaging filter. Because of the interaction between the two channels, $A(X)$ and $B(X)$ are defined recursively, based on two other functions $\mathcal{A}$ and $\mathcal{B}$:

$$\begin{aligned} A(X) &= \mathcal{A}(X, A(X), B(X)) \\ B(X) &= \mathcal{B}(X, A(X), B(X)). \end{aligned} \tag{3.2}$$

This recursion represents an interaction that evolves in time. Therefore the two functions $A(X)$ and $B(X)$ are the stabilized values of two functions that are dependent also on the time variable:

$$\begin{aligned} A(X) &= \lim_{t \to \infty} A^t(X, t) \\ B(X) &= \lim_{t \to \infty} B^t(X, t). \end{aligned} \tag{3.3}$$

The functions $A^t(X,t)$ and $B^t(X,t)$ can be based on $\mathcal{A}$ and $\mathcal{B}$, if we include an interaction delay Δt between the two channels:

$$\begin{aligned} A^t(X,t) &= \mathcal{A}(X, A^t(X, t - \Delta t), B^t(X, t - \Delta t)) \\ B^t(X,t) &= \mathcal{B}(X, A^t(X, t - \Delta t), B^t(X, t - \Delta t)). \end{aligned} \tag{3.4}$$

The use of two channels instead of a single one allows **ANNIE** to detect intensity changes at different scales, as suggested by Marr [18, 19], because each channel is tuned for components in a different spatial frequency range. Each channel has edge detectors tuned for horizontal and vertical orientations. All edge detectors interact with other neighboring detectors to achieve channel correlation and edge relaxation. Zero-crossings between outputs of Mexican-hat filters, as presented in Chapter 2, are at the base of the edge detection method. Each channel described above has one array of Mexican-hat filters as the first stage; the size of each such array is proportional to the space constant of the respective channel.

Lateral inhibition (interaction between detectors) achieves both channel correlation and edge relaxation. For relaxation, edge detectors should be reinforced by neighboring detectors that are activated by the same edge. On the other hand, inhibition has to be caused by neighboring detectors that are activated by all other edges. To prevent the detection of edge interferences, detectors in Channel B have to be inhibited by edges detected in Channel A placed close enough to be a cause of interference, but far enough to consider that they are not the same as the edge detected in Channel B. Only absolute values of the neighboring detectors' outputs are used because edge interference can occur between edges of any directions of contrast. One-step recursion simulates lateral inhibition and offers a very coarse approximation of equations 3.3 and 3.4. In feed-forward neural networks like **ANNIE** recursion is done through multiple layers. Therefore, there are two layers that perform two different stages in edge detection: zero-crossing detection and edge identification.

Training uses pairs of images. Each pair consists of an original picture as a reference to which the output is compared, and of an input picture that is obtained by applying noise and/or blur to the original one. The criterion used for training is to minimize the mean square error of the output compared to the reference picture. All the units of the network are analog. The reference picture has continuous pixel values between -1 and 1, and the input picture has its values in a similar range (the difference comes from the added noise). Through training, **ANNIE** learns the shapes of the weighted averaging filter and the edge correction filters, and the rules for edge relaxation and channel interaction.

3.2 The Operation of ANNIE

Each layer can be associated with a picture of its results. The following description of the neural network uses these pictures to illustrate the results in all layers, for a sample input pattern shown in Figure 6. **ANNIE** builds the output picture by scanning it pixel by pixel. The hidden layers use the same method to process the

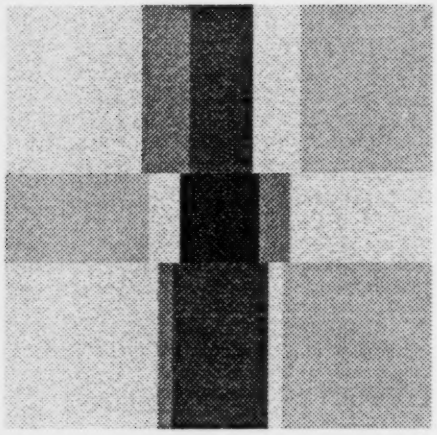


Figure 6: Sample input pattern. It has 175 × 175 pixels.

inputs from other layers. Only one unit is used in each layer to scan the results from inputting layers and to build a new picture of results. Although each layer contains only one unit used repeatedly while scanning, it is easier to understand the operation of the network if each layer is considered to be formed of an array of units, each corresponding to one pixel in the picture of results. In the network used for training, from lower layers to higher layers, the pictures of results are smaller, because units at the boundaries of the picture cannot have full connectivity to lower layers. After training, another network can be used, with all pictures of results having the same sizes as the input picture. In this case, units placed close to the boundaries would use only a partial connectivity, with weights derived from those used by units with full connectivity. However, the boundary pixels of the output picture will not be enhanced to the same degree as the other pixels.

Figure 7 illustrates the architecture of **ANNIE**. The input layer (0) conveys the degraded image to the network. The output layer (11) produces the processed picture and compares it to the reference picture. The hidden layers (1 to 10) constitute the edge detectors grouped into two channels, Channel A with a smaller space constant (for higher spatial frequency components) and Channel B with a larger space constant (for lower spatial frequency components). Each channel contains a layer (1 and 2, respectively) of Mexican-hat filters, two zero-crossings layers (3 and

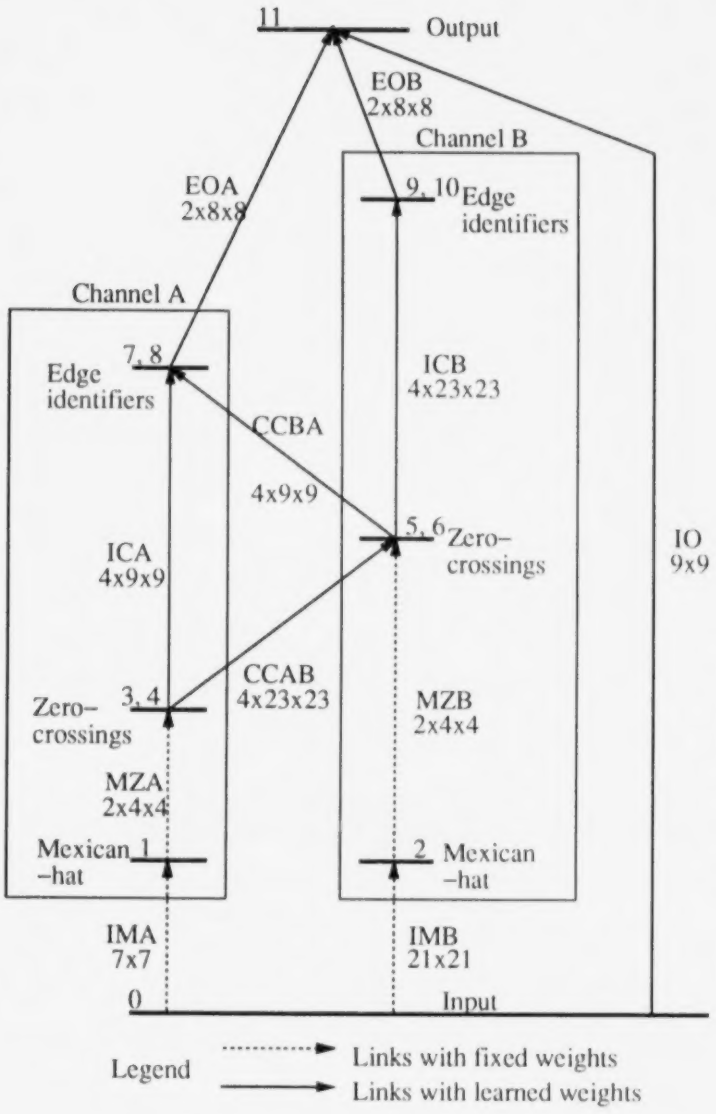


Figure 7: The architecture of **ANNIE**.

4, 5 and 6), and two edge identification layers (7 and 8, 9 and 10). The operation of the Mexican-hat filters could be executed by the zero-crossings layers; however, the Mexican-hat layers were used to separate the different functions in different layers and thus achieve modularity in the neural network. Edge detectors are duplicated into zero-crossings layers and edge identification layers to simulate recursive interaction in the feed-forward neural network; considering the Equations 3.4, zero-crossings layers simulate the values $A^t(X,0)$ and $B^t(X,0)$, while edge identification layers simulate the values $A^t(X,\Delta t)$ and $B^t(X,\Delta t)$. Units are linked to arrays of units from other layers. Therefore, links connecting one unit to a specific layer can be considered as being grouped in an array of links and their weights also as grouped in an array. These arrays will be referred as link arrays and weight arrays, respectively. Table 3.1 summarizes the links between layers. In Chapter 5 and in the Appendices, weights of all the links are illustrated with the values determined through training for different types of picture degradations.

All link arrays were chosen to be square to simplify the equivalence between vertical and horizontal edge detection. The larger the link arrays, the higher the accuracy in image enhancement would be. However, computational considerations restrict the sizes of the arrays. Apart from that, results are almost the same when these sizes are above a certain value, dependent on the spatial frequency of the channel. Edges detected in either channel can interfere only with edges situated at a distance equal to twice the space constant of the respective channel. Therefore, in the case of intra-channel and inter-channel links, arrays having the size equal to twice the space constant would be sufficient to correct edges affected by interference. After simulating the network with link arrays having the dimensions of the interference neighborhoods, smaller dimensions were chosen, in such a way that the speed of processing was increased without affecting the accuracy in a significant measure. For edge-detector – output and input – output links, different noise removal simulations were made first with large sizes and subsequently with smaller sizes, stopping when the accuracy was affected in a significant measure. Table 3.1 indicates the dimensions of the link arrays.

Category of links	Connection		Name and Dimensions
	From layer	To layer	
Input – Output	0	11	IO(9×9)
Input – Mexican-hat (IM)	0	1	IMA(7×7)
	0	2	IMB(21×21)
Mexican-hat – Zero-crossing (MZ)	1	3	MZA(4×4)
		4	
	2	5	MZB(4×4)
		6	
Intra-Channel (IC)	3	7	ICA(9×9)
	3	8	
	4	7	
	4	8	
	5	9	ICB(23×23)
	5	10	
	6	9	
	6	10	
Inter-Channel (Channel-Channel, CC)	3	5	CCAB(23×23)
	3	6	
	4	5	
	4	6	
	5	7	CCBA(9×9)
	5	8	
	6	7	
	6	8	
Edge-detector – Output (EO)	7	11	EOA(8×8)
	8		
	9		EOB(8×8)
	10		

Table 3.1: The links between **ANNIE**'s layers.

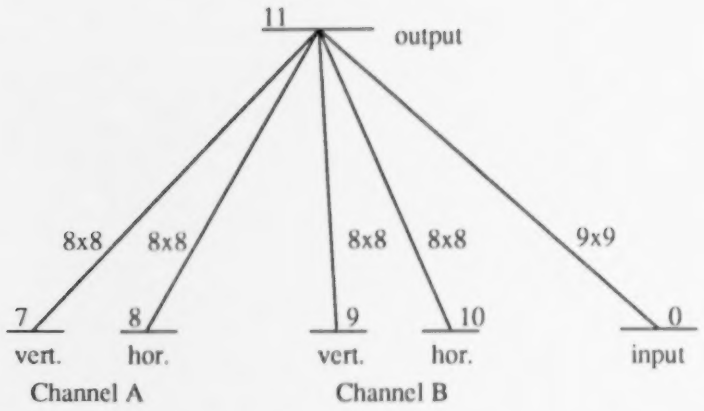


Figure 8: The links entering the output layer.

The weights of the links between the input layer and the Mexican-hat layers are determined by initial calculations to simulate the shape of the Mexican-hat filters. These weights are not changed during the training process. The weights of the links between the Mexican-hat layers and the zero-crossing layers are not used, and for this reason, they also are not changed during the training process. Some weights are expected to have a specific sign and some weight arrays are expected to be symmetrical; some arrays are expected to be the transpose of others. These features are reinforced during training, as shown in Section 4.4.

3.2.1 The Operation of the Output Layer

To calculate the value of each pixel, the output unit of the network makes a weighted average of the input units in a 9×9 neighborhood and adds to this result edge corrections from the two edge-detection channels. There are two edge identification layers at the top of each channel: one layer is tuned for vertical edges, the other layer is tuned for horizontal edges. Diagonal edges are detected by both layers and are sharpened with a sum of both layers' outputs. The output unit receives inputs from arrays (8×8) of units in the top two layers of each channel. Figure 8 shows how the output layer is connected and from which layers it receives inputs. The

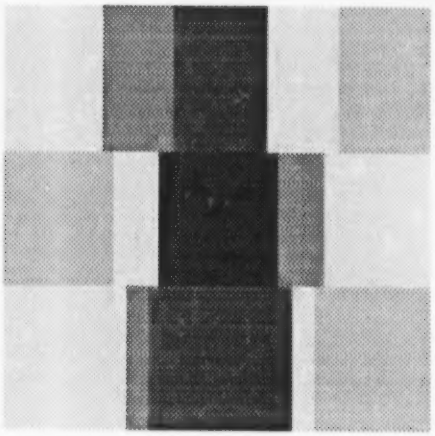


Figure 9: The output for the sample picture. It has 115×115 pixels.

operation of the output layer is linear; for each pixel, its value in the output picture is:

$$o = \sum_{i}^{9\times9} w_i^{oa} y_i^a + \sum_{i}^{8\times8} w_i^{oA,v} y_i^{A,v} + \sum_{i}^{8\times8} w_i^{oA,h} y_i^{A,h} + \sum_{i}^{8\times8} w_i^{oB,v} y_i^{B,v} + \sum_{i}^{8\times8} w_i^{oB,h} y_i^{B,h} \quad (3.5)$$

where y_i^a are the inputs from the input layer, $y_i^{A,v}$ and $y_i^{A,h}$ are the inputs from the vertical and horizontal detectors in Channel A, $y_i^{B,v}$ and $y_i^{B,h}$ are the inputs from the vertical and horizontal detectors in Channel B, and w_i^{oa}, $w_i^{oA,v}$, $w_i^{oA,h}$, $w_i^{oB,v}$, and $w_i^{oB,h}$ are the weights of the corresponding links. The first term of the sum represents the operation of weighted averaging and the other four terms represent the edge corrections. The weights w_i^{oa} implement the filter illustrated in Figure 4, while the weights $w_i^{oA,v}$, $w_i^{oA,h}$, $w_i^{oB,v}$, and $w_i^{oB,h}$ implement four filters like the one illustrated in Figure 5. Figure 9 presents the output of **ANNIE** for the sample picture shown in Figure 6.

3.2.2 The Operation of the Edge Detection Channels

Each channel executes three operations:

1. It applies a Mexican-hat operator with a specific size over the input picture.

2. It detects zero-crossings in the pictures resulting from the first operation.

3. Considering each zero-crossing as an edge, it removes or at least attenuates false edges caused by noise or interference.

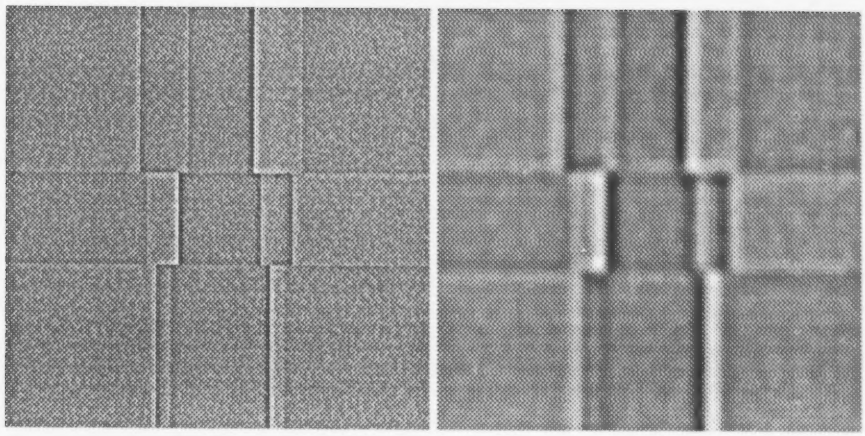
The Mexican-hat Layers

In each channel, a first layer (layers 1, 2 in Figure 7) applies a Mexican-hat operator over the input picture. Each unit in these layers is connected to an array of units from the input layer. The weights of these connections determine the shapes of the Mexican-hat operators. The size of the array is proportional to the space constant of the respective channel. For Channel A, the array is 7×7: a 3×3 array of weights in the center are positive, while the others are negative. These are the minimal dimensions with which a Mexican-hat filter can achieve smoothing before applying the Laplacian operator. For Channel B, the array is 21×21. This dimension, while relatively still small, allows Channel B to be significantly less sensitive to noise than Channel A. Campbell and Robson [10] stated that mechanisms in the visual nervous system respond maximally at some particular spatial frequency and hardly at all at spatial frequencies differing by a factor of two. This suggests that a system like **ANNIE** should use channels having space constants differing at most by a factor of two. But because only two such channels are used, a factor of three was preferred to illustrate better the differences and the interaction between the channels. Figure 10 illustrates the results of processing the sample picture with each of the two Mexican-hat operators having different space constants.

Equation 2.1 determines the weights of the links coming into the units of these layers, with two changes:

1. The weight of the central connection is modified such that the sum of all the weights is null. This ensures that the response of a Mexican-hat unit is zero over an array of equal input values.

2. All the weights are normalized such that the sum of the negative ones and the sum of the positive ones are both equal in absolute value to 1, in both channels.



(a) (b)

Figure 10: The sample picture processed by Mexican-hat layers. (a) processed by Channel A's 7×7 Mexican-hat layers (169×169 pixels). (b) processed by Channel B's 21×21 Mexican-hat layers (155×155 pixels).

> This ensures that the responses of Mexican-hat units in both channels are of the same order of magnitude for a particular edge. Without this restriction, training would erroneously favor the channel with a higher response.

Training these weights can improve the performance of the network, but affects negatively the learning speed. Therefore, they are calculated when the network is built and are kept with the same values while the network is trained. A factor in optimizing the Mexican-hat units is choosing a value for σ, the space constant in Equation 2.1. Through trial-and-error, σ was chosen as 1/8.5 the size of the link array. This value is roughly the best one for which the distribution of the weights reproduces closely enough the shape presented in Figure 1.

The Zero-crossing Layers

Each Mexican-hat layer sends its outputs to two other layers (3, 4 and 5, 6 in Figure 7), one tuned to detect vertical zero-crossings and one tuned to detect horizontal zero-crossings in the picture resulting from the Mexican-hat filtering. Choosing a zero-crossing operator was a difficult task, because of several requirements whose

p_1	p_2	p_3	p_4
p_5	p_6	p_7	p_8
p_9	p_{10}	p_{11}	p_{12}
p_{13}	p_{14}	p_{15}	p_{16}

Figure 11: The array of outputs of the Mexican-hat layers, used for zero-crossing detection.

incomplete fulfillment is the most serious source of errors:

- Linearity of the measurement operation's characteristic: the output should be proportional to the contrast magnitude of the edge.
- Thinness of edges: edges should be detected as lines only one or at most two pixels thick, according to whether the edge is placed between or over pixels; when edges are detected as lines two pixels thick, measurement of the contrast magnitude should be correlated with the effect that a thick line has on the edge correction operation.
- Simplicity: operators used for different orientations should be as similar as possible, to keep the program simple and easy to maintain.

Units in these layers are connected to an array of 4×4 units in the Mexican-hat layer. The size of the operator was chosen on the consideration that a larger one, although less sensitive to noise, would have omitted more edge pixels. This factor is more important than noise because edge pixels caused by noise are removed or at least attenuated by edge relaxation.

The weights of the corresponding links have no significance; only the outputs p_1 to p_{16} of the units are used as presented in Figure 11. Two values, v_1 and v_2 are calculated. A zero-crossing is detected if $v1$ and $v2$ are of opposite signs. The amplitude of the zero-crossing is

$$z = \sqrt{-(v_1 \cdot v_2)}. \tag{3.6}$$

From Figure 11, for vertical detectors

$$v_1 = p_1 + p_2 + p_5 + p_6 + p_9 + p_{10} + p_{13} + p_{14} \tag{3.7}$$

and

$$v_2 = p_3 + p_4 + p_7 + p_8 + p_{11} + p_{12} + p_{15} + p_{16}. \tag{3.8}$$

For horizontal detectors

$$v_1 = p_1 + p_2 + p_3 + p_4 + p_5 + p_6 + p_7 + p_8 \tag{3.9}$$

and

$$v_2 = p_9 + p_{10} + p_{11} + p_{12} + p_{13} + p_{14} + p_{15} + p_{16}. \tag{3.10}$$

Figure 12 illustrates how zero-crossings are detected in both channels for the sample picture. While Channel A is more sensitive to noise, Channel B is more sensitive to edge interferences. If edges are close to each other, they can be detected as displaced or as only one edge, or they can cause a false edge in between.

Lateral Inhibition and the Edge Identification Layers

If we consider that the two channels are formed of edge detectors with vertical and horizontal orientations, edge relaxation and inter-channel correlation are achieved by simulated lateral inhibition from edge detectors of both orientations. To achieve a lateral inhibition effect, every edge detector i should be connected to detectors from both channels, placed in a neighborhood $N(A)$ and in a neighborhood $N(B)$, respectively, around pixel i, including pixel i in the other channel. With the amplitude of the zero-crossing z_i defined by Equation 3.6, y_i, the output of edge detector i, should satisfy the relation:

$$y_i = sgn(z_i) \cdot pos\left(|z_i| + \sum_{j}^{N(A)\cup N(B)} w_{ij} \cdot |y_j|\right), \tag{3.11}$$

where

$$sgn(x) = \begin{cases} -1 & \text{if } x < 0 \\ 0 & \text{if } x = 0 \\ 1 & \text{if } x > 0 \end{cases} \quad \text{and} \quad pos(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}. \tag{3.12}$$

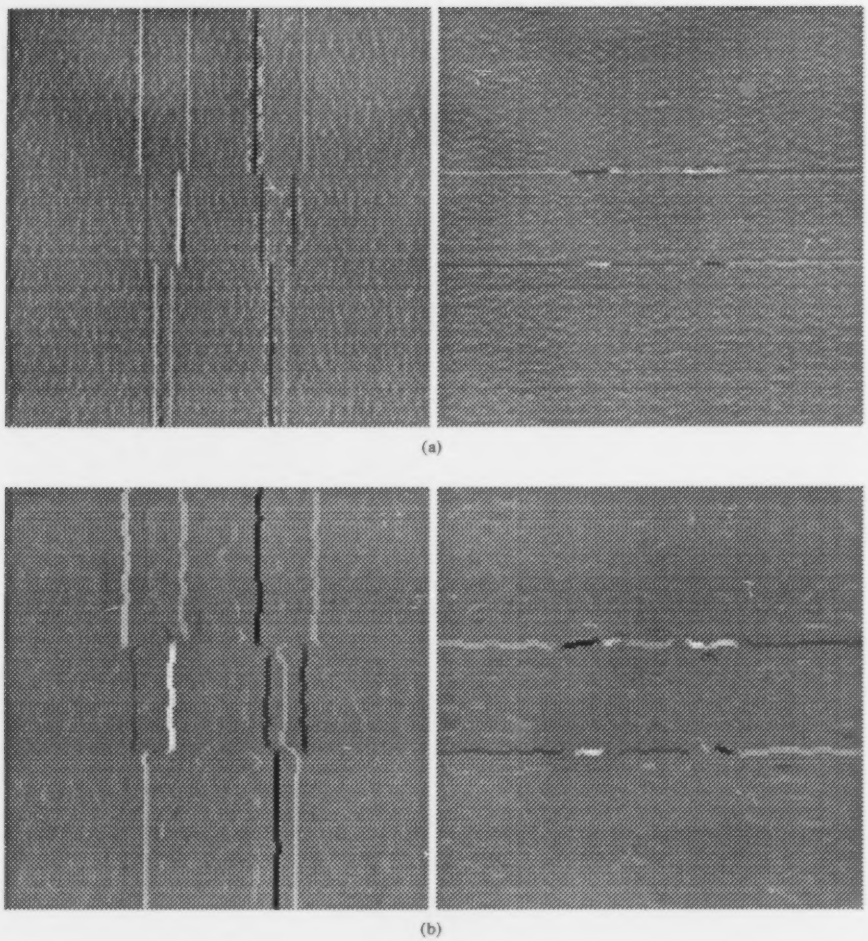


Figure 12: Zero-crossings detected in both channels for the sample picture. (a) vertical and horizontal zero-crossings detected by Channel A. Both pictures have 166 × 166 pixels. (b) vertical and horizontal zero-crossings detected by Channel B. Both pictures have 144 × 144 pixels.

y_j are outputs of the edge detectors connected to unit i, and w_{ij} are the weights of the corresponding links. With this relation, lateral inhibition is not allowed to create edges by itself or to reverse the direction of contrast for a detected zero-crossing. It can only reinforce or attenuate edges already detected. The function in Equation 3.11 is recursive because the outputs y_j are also dependent on y_i. Because the neural network is a feed-forward one and units cannot be connected to units from the same layer or a lower layer, the function cannot be implemented as such. Instead, it can be approximated by a finite iteration:

$$y_i^{(t)} = sgn(y_i^{(t-1)}) \cdot pos\left(\left|y_i^{(t-1)}\right| + \sum_j^{N(A)} w_{ij}^A \cdot \left|y_j^{A(t-1)}\right| + \sum_j^{N(B)} w_{ij}^B \cdot \left|y_j^{B(t-1)}\right|\right), \quad (3.13)$$

where

$$y_i^{(0)} = z_i. \quad (3.14)$$

Only two steps of this iteration are implemented in **ANNIE**. Using more steps would improve the performance but would also make the network more complex. Every edge detector is duplicated as a zero-crossing unit and an edge identification unit pair; the two members of a pair are connected but the weight of the link between them is not used. The edge identification layers are connected to the zero-crossing layers and only in Channel B the zero-crossing layers are connected to the zero-crossing layers of Channel A. The zero-crossing layers execute the first iteration step, by calculating the zero-crossing intensities z_i as in Equation 3.6. The outputs of these layers in Channel A are:

$$y_i^{A(0)} = z_i^A. \quad (3.15)$$

In Channel B, units in the zero-crossing layers are connected to a neighborhood NB(A) of units in Channel A's zero-crossing layers and the outputs are

$$y_i^{B(0)} = sgn(z_i^B) \cdot pos\left(\left|z_i^B\right| + \sum_j^{NB(A)} w_{ij}^{BA} \cdot \left|y_j^{A(0)}\right|\right) \quad (3.16)$$

because the outputs $y_{j(A)}^{(0)}$ are already available. The edge identification layers execute the second iteration step. In Channel A, the units in these layers are connected

to a neighborhood NA(A) of units in Channel A's zero-crossing layers and to a neighborhood NA(B) of units in Channel B's zero-crossing layers. The outputs are given by:

$$y_i^{A(1)} = sgn(y_i^{A(0)}) \cdot pos\left(\left|y_i^{A(0)}\right| + \sum_{j \neq i}^{NA(A)} w_{ij}^{AA} \cdot \left|y_j^{A(0)}\right| + \sum_{j}^{NA(B)} w_{ij}^{AB} \cdot \left|y_j^{B(0)}\right|\right) \quad (3.17)$$

In Channel B, the units in the edge identification layers are connected to a neighborhood NB(B) of units in Channel B's zero-crossing layers and the outputs are

$$y_i^{B(1)} = sgn(y_i^{B(0)}) \cdot pos\left(\left|y_i^{B(0)}\right| + \sum_{j \neq i}^{NB(B)} w_{ij}^{BB} \cdot \left|y_j^{B(0)}\right|\right) \quad (3.18)$$

Edge identification units in Channel A are connected to arrays of 9×9 units in the zero-crossing layers of both orientations, from both channels. Edge identification units in Channel A are connected to arrays of 23×23 units in the zero-crossing layers of both orientations from the same channel. Zero-crossing units in Channel B are connected to arrays of 23×23 units in the zero-crossing layers of both orientations from Channel A. Figure 13 illustrates the connectivity inside and between the two channels. Unlike in Figure 7, the zero-crossing and edge identification layers are shown separately as vertical and horizontal detector layers, and the links between them are also shown separately. Figure 14 illustrates the results from the edge identification layer for the sample picture shown in Figure 6. The effects of lateral inhibition can be detected by comparing these pictures to the ones in Figure 12.

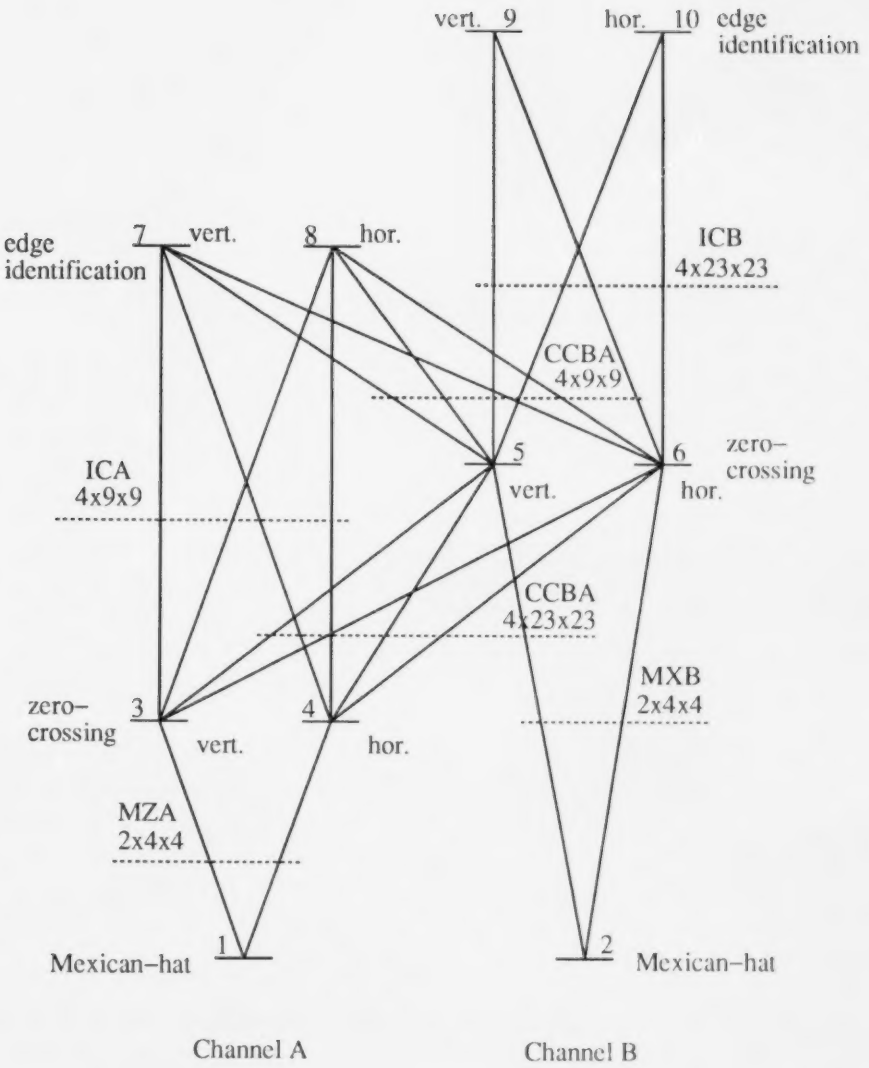


Figure 13: The links inside and between the two channels.

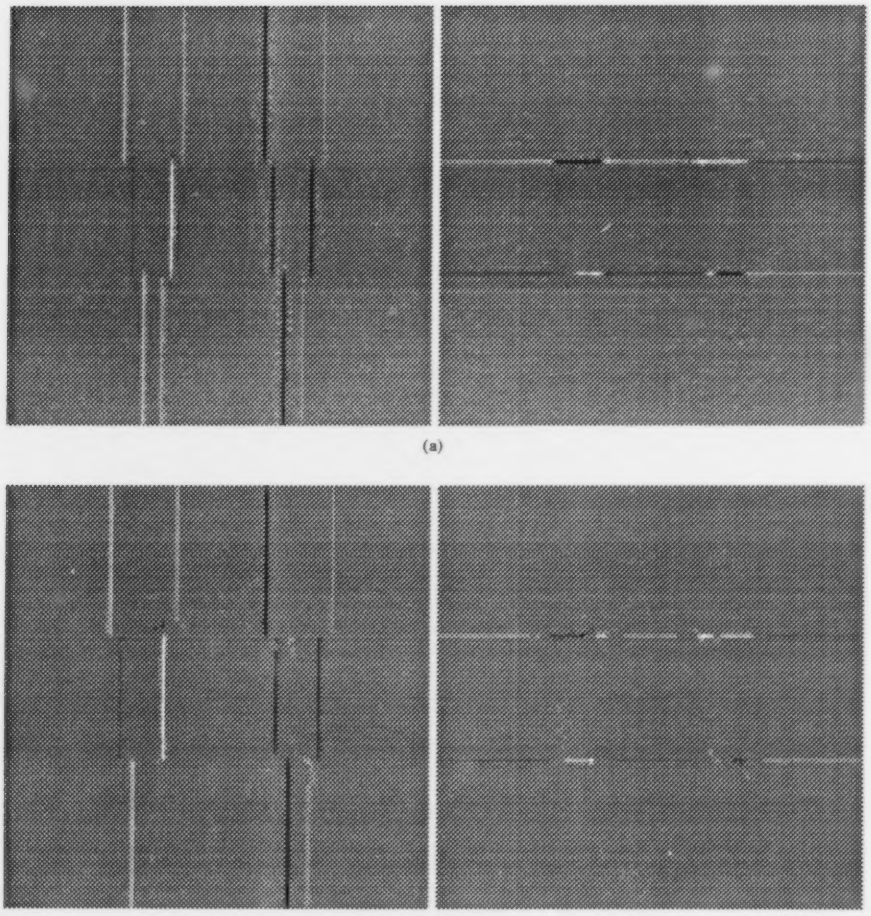


Figure 14: Edges identified in both channels for the sample picture. (a) vertical and horizontal edges identified by Channel A. Both pictures have 136 × 136 pixels. (b): vertical and horizontal edges identified by Channel B. Both pictures have 122 × 122 pixels.

Chapter 4

The Training Method

The algorithm used for training is quickprop [32], a derivation of the backpropagation algorithm [13]. There were several reasons behind this choice:

1. **ANNIE** is a neural network with continuous-valued outputs.

2. A supervised procedure is necessary for this task. An unsupervised procedure cannot guarantee that of all possible image processing operations, the network will learn the ones of noise removal and edge sharpening. A reinforcement procedure needs a quality measure, and such a measure is hard to find for the task of image enhancement.

3. The complexity of edge sharpening requires hidden layers in the neural network. The hidden layers are used for edge detection and the information they provide is used to correct the blur. Without the hidden layers, the neural network cannot learn to correct the blur.

4. For most applications, quickprop is much faster than conventional backpropagation or other backpropagation derivations, and has a reduced computational complexity.

Weights are updated not in an on-line mode, but in a batch mode. In both modes, during one training cycle, the values of all the output pictures' pixels are calculated.

For each pixel, a correction is calculated for each weight in the network. In an on-line mode, weights are changed after each output pixel calculation, by their determined corrections. In a batch mode, weights are changed by the average of all their corrections determined during the entire training cycle. This ensures a smoother search for the optimal solution. An on-line method can be faster, but for a complex network it can cause instability.

4.1 Overview of Backpropagation

The backpropagation training algorithm [13, Chapter 8] is an iterative gradient algorithm designed to minimize the mean square error between the actual output of a multilayer feed-forward neural network and the desired output. Training through backpropagation corresponds to performing gradient descent on a surface in weight space whose height at any point in weight space is equal to the error measure. This surface is called the "error surface."

Let the measure of the network's error to input pattern p from among P input patterns be given by the mean square sum

$$E_p = \frac{1}{2} \sum_j (d_{pj} - o_{pj})^2 , \tag{4.1}$$

where o_{pj} is the actual output of output unit j and d_{pj} is its desired output. Then, the overall measure of error is $E = \sum_p E_p$. In batch mode training, minimizing this error through gradient descent can be achieved by starting with any set of weights and repeatedly changing each weight w_{ij} from unit i to unit j by an amount proportional to $\partial E / \partial w_{ij}$:

$$\Delta w_{ij} = -\frac{\eta}{P} \cdot \sum_p \frac{\partial E_p}{\partial w_{ij}}, \tag{4.2}$$

where η is the constant of proportionality representing the learning rate. To calculate $\partial E_p / \partial w_{ij}$, we can write

$$\frac{\partial E_p}{\partial w_{ij}} = \frac{\partial E_p}{\partial o_{pj}} \cdot \frac{\partial o_{pj}}{\partial w_{ij}}. \tag{4.3}$$

If we define

$$\delta_{pj} = -\frac{\partial E_p}{\partial o_{pj}},$$

there are two cases for calculating δ_{pj} [1]:

1. For an output unit of the network

$$\delta_{pj} = (d_{pj} - o_{pj}) \tag{4.4}$$

2. For a hidden unit of the network

$$\delta_{pj} = \sum_k \delta_{pk} \cdot \frac{\partial o_{pk}}{\partial o_{pj}}, \tag{4.5}$$

where o_{pk} are the outputs of the units for which o_{pj} is an input.

Rumelhart, Hinton, and Williams [13, Chapter 8] show how the backpropagation procedure can be applied to recurrent networks. A network in which the states of the units at time t determine the states of the units at time $t+1$ is equivalent to a net that has one layer for each time slice. Each weight in the recurrent network is implemented by a whole set of identical weights in the corresponding layered net, one for each time slice.

4.2 Quickprop as an Improvement to Backpropagation

Despite its impressive performance on small problems, and its promise as a widely applicable mechanism, backpropagation is inadequate for larger tasks because the

[1] Usually [13, Chapter 8], backpropagation is described slightly differently, by defining $\delta_{pj} = -\frac{\partial E_p}{\partial net_{pj}}$, where net_{pj} is the net input of the unit j for input/output pair p. This difference exists because usually, o_{pj} is a function of net_{pj} and $net_{pj} = \sum_i w_{ji} o_{pi}$. For **ANNIE**, o_{pj} are directly a function of w_{ji} and o_{pi}. That is why the above description was preferred, although it is equivalent to the usual one.

learning time scales poorly. Two approaches to the problem of increasing the learning speed have been tried. The first approach is to adjust the learning rate dynamically, either globally or separately for each weight, based on the history of the learning procedure. The momentum term used in standard backpropagation [13, Chapter 8] is a form of this strategy; Jacobs [33] has proposed heuristics that allow every weight to have its own adjustable learning rate. The other approach makes explicit use of the second derivative of the error with respect to each weight. Given this information, we can select a new set of weights using Newton's method or some more sophisticated optimization technique. Unfortunately, it requires a costly global computation to derive the true second derivative, so some approximation is used. Parker [34], Watrous [35], and Becker and LeCun [36] have all been active in this area.

Quickprop, developed by Fahlman [32], is a second-order method, based loosely on Newton's method. Everything proceeds as in standard backpropagation, but for each weight a copy is kept of $(\partial E/\partial w)(t-1)$, the error derivative computed during the previous training cycle, along with the previous change of the weight. The $(\partial E/\partial w)(t)$ value for the current training cycle is also available at weight-update time.

Quickprop is based on two assumptions: first, that the error versus weight curve for each weight can be approximated by a parabola whose arms open upward; second, that the change in a weight does not affect the slope of the error curve seen by other weights. For each weight, independently, the procedure uses the previous and current error slopes to determine a parabola. Then, considering the last weight change, training changes the weight to the minimum point of this parabola. The computation is made according to the formula:

$$\Delta w(t) = \frac{S(t)}{S(t-1) - S(t)} \cdot \Delta w(t-1), \tag{4.6}$$

where $S(t)$ and $S(t-1)$ are the current and previous values of $\partial E/\partial w$.

A difficulty occurs when the current slope is in the same direction as the previous slope, but is the same size or larger in magnitude. Applying Equation 4.6 blindly

would cause taking an infinite step or moving uphill. In these cases, the new step is equal instead to μ times the previous step, where μ is a parameter called the maximum growth factor. Besides the cases mentioned before, no weight step is allowed to be greater in magnitude than μ times the previous step for that weight. According to Fahlman, a value of 1.75 for μ works well for a wide range of problems. In some simulations, this value has caused **ANNIE** to oscillate when close to the convergence point. In these cases, smaller values for μ were used, down to 1.1.

Since quickprop changes weights based on what happened during the previous weight update, a way to bootstrap the process is necessary. The solution is to use gradient descent, based on the current slope and a learning rate η, to start the process. This method is used also to restart the process for any weight that has a previous step size of zero. Because η is mostly used only when the simulation is started, its value is not of high importance. η was therefore chosen to be 0.001, small enough not to cause instability in any simulation.

4.3 Error Backpropagation and Weight Changes for ANNIE

This section shows in more detail how Equations 4.2, 4.4, 4.5, and 4.6 are applied in training **ANNIE**. The elements necessary for calculating weight changes are the values δ_{pj} from Equations 4.4 and 4.5, and the values $S(t) = \partial E/\partial w$ from Equation 4.6. Training changes only the links going into the output unit and the links implementing lateral inhibition.

To eliminate the division by the number of output pixels P in Equation 4.2 (used for every layer), Equation 4.4 was changed into

$$\delta_i^o = \frac{d_i - o_i}{P},$$

for each output pixel i. Then for any link to the output layer, its weight is changed according to

$$\frac{\partial E}{\partial w_{ij}^o} = \sum_i \delta_i^o \cdot y_j^{(i)}.$$

where $y_j^{(i)}$ is the input that affects output pixel i combined with the weight w_{ij}^o.

For all the edge identification layers, the error propagated to each pixel is

$$\delta_i^{(1)} = \sum_j \delta_j^o \cdot w_{ij}^o,$$

where δ_j^o is the error in the output pixel connected to the edge identification pixel i by a link with the weight w_{ij}^o. With the propagated errors $\delta_i^{(1)}$, the weights of the links going into edge identification layers are changed respectively according to

$$\frac{\partial E}{\partial w_{ij}^{AA}} = \sum_i \delta_i^{A(1)} \cdot \left| y_j^{A(0)} \right|,$$

$$\frac{\partial E}{\partial w_{ij}^{AB}} = \sum_i \delta_i^{A(1)} \cdot \left| y_j^{B(0)} \right|,$$

and

$$\frac{\partial E}{\partial w_{ij}^{BB}} = \sum_i \delta_i^{B(1)} \cdot \left| y_j^{B(0)} \right|,$$

where $y_j^{A(0)}$ and $y_j^{B(0)}$ are respectively the inputs from the first and second channel that affect edge identifictaion pixel i combined with the weight w_{ij}^{AA}, w_{ij}^{AB}, or w_{ij}^{BB}, and $\delta_i^{A(1)}$ and $\delta_i^{B(1)}$ are the errors propagated to the edge identification layers.

For the zero-crossing layers in Channel B, the error propagated to each pixel is

$$\delta_i^{B(0)} = \delta_i^{B(1)} + sgn(y_i^{B(0)}) \cdot \left(\sum_j \delta_j^{A(1)} \cdot w_{ij}^{AB} + \sum_j \delta_j^{B(1)} \cdot w_{ij}^{BB} \right),$$

where $y_i^{B(0)}$ is the value of pixel i in the zero-crossing layer of channel B, $\delta_i^{B(1)}$ is the error propagated to pixel i in the edge identification layer, $\delta_j^{A(1)}$ and $\delta_j^{B(1)}$ are the errors propagated to pixels j in the edge identification layers of channel A and channel B, and w_{ij}^{AB} and w_{ij}^{BB} are the weights connecting zero-crossing pixel i to the respective pixels j in the edge identification layers of channel A and channel B respectively. With the propagated errors $\delta_i^{B(0)}$, the weights of the links going into zero-crossing layers of Channel B are changed according to

$$\frac{\partial E}{\partial w_{ij}^{BA}} = \sum_i \delta_i^{B(0)} \cdot \left| y_j^{A(0)} \right|,$$

where $y_j^{A(0)}$ are the inputs from the first channel that affect output pixel i combined with the weight w_{ij}^{BA}.

4.4 Constraints in Updating Weights

We expect the weight arrays to have some characteristics after training:

- Some weight arrays should have one, two, or four axes of symmetry.
- Some of these weight arrays should be the transpose of other arrays.
- Some weights should have a certain sign.
- In some arrays, there are weights that should be equal in absolute value, but of opposite sign.

These constraints can be forced during training. Forcing them has the advantage of increasing the learning speed because it does not allow the search to go in a wrong direction. It also prevents the network from loosing its generality and becoming trained to features specific to the training pattern.

Let us use the notation w_{ij} for all the weights with the meaning that they have the position *(i, j)* in an array of dimension s. In the weighted averaging of the input picture, pixels placed at the same distance from the center should have the same weight, indifferent to their orientation to the center. Therefore, the weights of the input – output links should form an array symmetric to the center. This can be achieved only by complex calculations and instead, a more relaxed, four-axes symmetry is used, and the weights of the input–output links have to satisfy the following constraints:

$$w^a_{ij} = w^a_{(s-i)j} = w^a_{i(s-j)} = w^a_{(s-i)(s-j)} = w^a_{ji} = w^a_{(s-j)i} = w^a_{j(s-i)} = w^a_{(s-j)(s-i)}. \quad (4.7)$$

Edge identification–output link weights that are symmetrical to the axis with the orientation for which the detectors are tuned have to be equal in absolute value but with opposite signs. Weight arrays that are used by horizontal edge detectors must be the transpose of weight arrays used by vertical edge detectors. Therefore, for each channel, the weights between the edge identification layers and the output

layer are forced to satisfy the following constraints:

$$w^v_{ij} = -w^v_{i(s-j)} = w^v_{(s-i)j} = -w^v_{(s-i)(s-j)} = w^h_{ji} = -w^h_{j(s-i)} = w^h_{(s-j)i} = -w^h_{(s-j)(s-i)}, \tag{4.8}$$

where w^v are weights of links coming from vertically tuned layers and w^h are weights of links coming from horizontally tuned layers. In the same manner, the weight arrays implementing lateral inhibition in horizontal detectors must be the transpose of weight arrays in vertical detectors. These weights, identified in Equations 3.16–3.18 as w^{AA}, w^{AB}, w^{BA}, and w^{BB} have to satisfy separately the following constraints:

$$w^{vh}_{ij} = w^{vh}_{(s-i)j} = w^{vh}_{i(s-j)} = w^{vh}_{(s-i)(s-j)} = w^{hv}_{ji} = w^{hv}_{(s-j)i} = w^{hv}_{j(s-i)} = w^{hv}_{(s-j)(s-i)} \tag{4.9}$$

and

$$w^{vv}_{ij} = w^{vv}_{(s-i)j} = w^{vv}_{i(s-j)} = w^{vv}_{(s-i)(s-j)} = w^{hh}_{ji} = w^{hh}_{(s-j)i} = w^{hh}_{j(s-i)} = w^{hh}_{(s-j)(s-i)}, \tag{4.10}$$

where w^{vh}, w^{hv}, w^{vv}, and w^{hh} are respectively weights of links from vertical to horizontal layers, from horizontal to vertical layers, from vertical to vertical layers, and from horizontal to horizontal layers. To implement weighted averaging, the input–output link weights are forced to be positive.

To force these constraints on weights that have to be equal, their changes are averaged together and then applied to all the corresponding weights. Consequently, during the training of the neural network, information has to be exchanged between units. This communication represents a violation of the locality of processing constraint and therefore limits the implementability of the neural network in a massively parallel architecture. However, a massively parallel implementation is possible after training is finished. A certain degree of parallelism is achievable even while training, and a parallel implementation can offer better performance.

Chapter 5

Simulations

Through training, **ANNIE** has the ability to learn how to eliminate noise and how to sharpen edges. It generalizes this knowledge to enhance pictures that have the same type of noise and the same type of blur as the pictures used for training. To prove this ability, **ANNIE** was trained with nine pictures and then was tested on a tenth. Figure 15 shows the nine pictures used for training and Figure 16 shows the picture used for testing. All these pictures are artificially generated. The original pictures have no noise and no blur and are used as reference patterns at the output of the network. The same pictures degraded by noise and/or blur are used as input patterns. During training, **ANNIE** processes the input patterns and compares the results to the reference patterns. The differences between the output and the reference are used for changing the weights of the network. Simulations were performed for several types of image degradation: noise and blur, only blur, and only noise. This chapter (Section 5.2) presents only the results for removing both noise and blur. Appendix A presents the results for edge sharpening in the presence of noise, Appendix B presents results for removing noise, and Appendix C presents results for edge sharpening. Appendix D illustrates the results obtained on digitized real pictures and a short comparison between **ANNIE** as a noise removal method and median filtering.

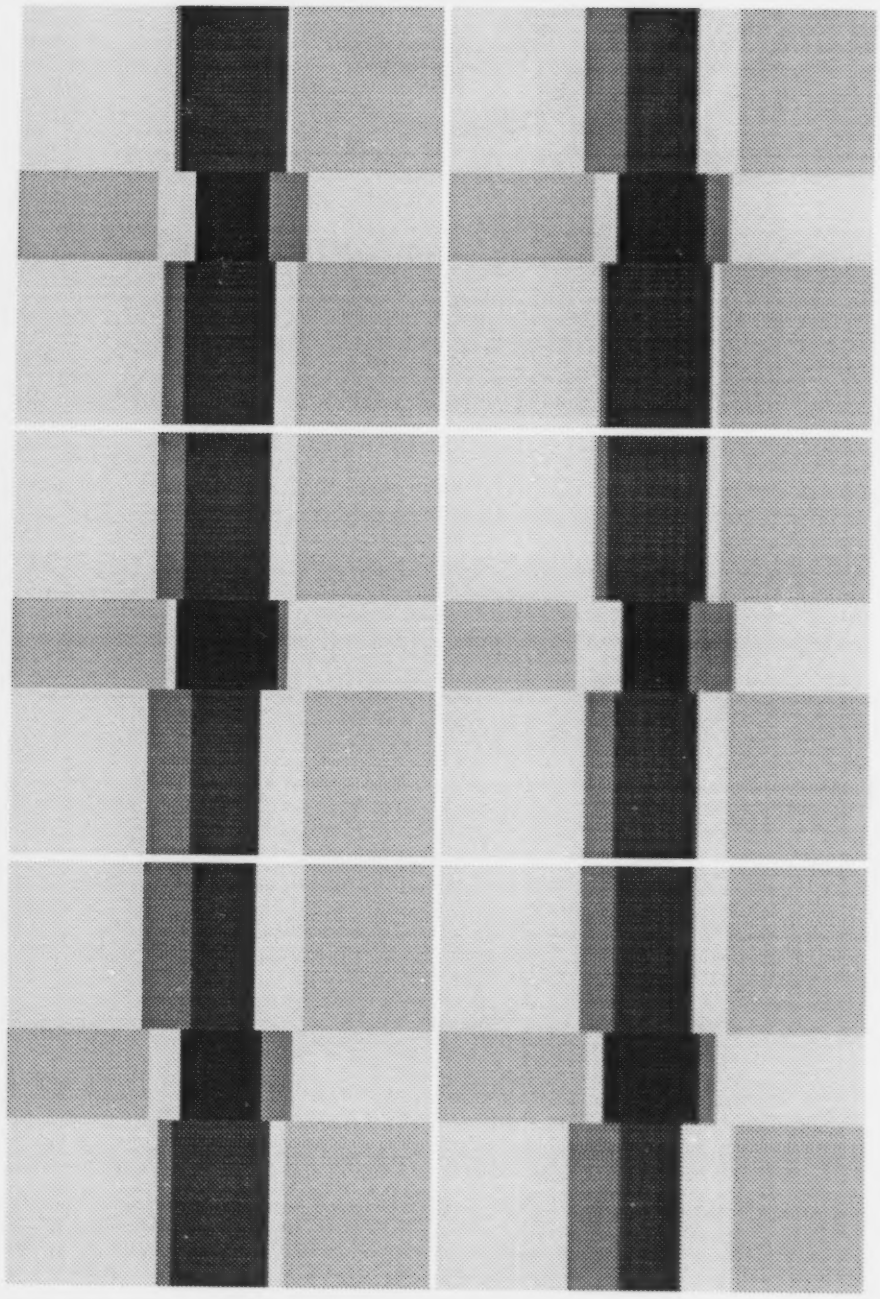


Figure 15: The pictures used for training.

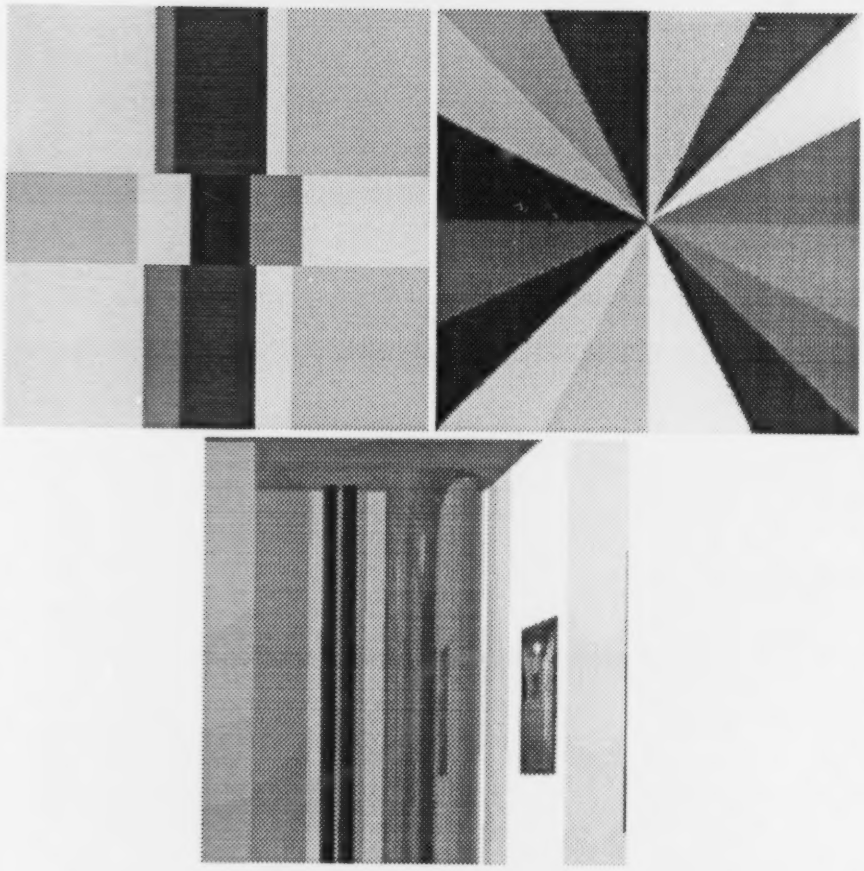


Figure 15[Cont.]: The pictures used for training.

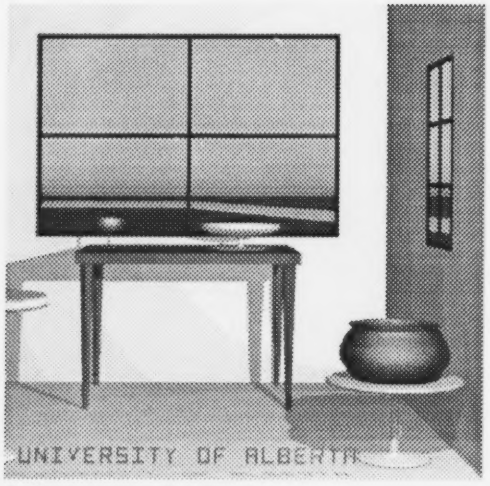


Figure 16: The picture used for testing.

5.1 Methodology of Simulation

Performance Criterion

The criterion for judging the performance of **ANNIE** is the mean square error $E = \sum_j (d_j - o_j)^2$, calculated for all the pixels j in a set of patterns. This error is minimized during training for the set of training patterns. The final error has to be judged by comparing it to the initial error, which represents the difference between the input pattern and the reference pattern. For the training patterns, the final mean square error represents the limitations of the system. The error cannot be eliminated completely because:

- Not all edges can be detected even by the channel with a smaller spatial constant because of edge interference.
- Not all false edges and not all edge pixels caused by noise can be eliminated.
- There is no edge correction that is perfect for edges of all orientations and placements relative to the picture pixels.
- There is always a tradeoff between noise removal and edge sharpening.

For the test pattern, the proportion in which the mean square error is reduced shows the power of the system to generalize the knowledge acquired during training. For a good performance, the training set has to include all the significant features. Even so, the neural network makes a tradeoff between averaging for noise elimination on one hand, and edge correction on the other. That is why the proportion of edge pixels affects the optimization of the mean square error. If the training patterns have only few edges, noise removal will be emphasized to the detriment of deblurring; the opposite is also true. Therefore, results are not optimal for the test pattern if it has a proportion of edge pixels different from the one in the trainig set.

Training Patterns

There are several features that **ANNIE** must learn to discern. The training patterns attempt to include all these features:

- Parallel edges placed at different distances from each other.
- Edges oriented at different angles.
- One-pixel and two-pixel wide edges.
- Edges of different contrast magnitudes.

Two edges interfere in channel A if they are placed at a distance smaller than 8 pixels from each other, and in channel B at a distance smaller than 22 pixels. To train **ANNIE** to prevent such interferences, the first seven training pictures have parallel edges placed at distances from 2 to 22 pixels. The eighth and the ninth training pictures have edges oriented at different angles. Half of the first eight pictures have one-pixel wide edges and the other half have two-pixel wide edges. Two-pixel wide edges are generated from one-pixel wide edges by blurring them through an averaging 2×2 matrix. The ninth training picture is used especially because it is generated with characteristics similar to those of the test picture.

Internal Representation and External Visualization of Pictures

The pictures are represented by arrays of pixels. When running a simulation of **ANNIE**, the pictures are read from separate files. In these files, pixels are represented by integer values between 0 and 255. Once read by the program, pixels are represented by floating point variables that take values between -1.0 and 1.0. The values written into the input layer and written out by the output layer can be outside this range because of the added noise. The conversion is made by mapping the [0, 255] range directly into the range [-1.0, 1.0]. This internal normalization is useful because training of the averaging links is faster when there are both negative and positive values. Internal pictures in hidden layers take values in different ranges. These pictures are normalized to the range [0, 255] when printed for visualization. All pictures presented in this thesis are executed on laser printers that have only 16 grey levels. Therefore, gradual variations in the grey intensity may appear in print as sudden variations (apparent edges), because 16 small steps on the 0 to 255 grey scale are represented by one large, single step on the 0 to 15 scale.

Picture Degradations

Blur was generated by replacing the value of each pixel with the average of the pixels in a 3×3 neighborhood. In most simulations, noise was generated by adding random values uniformly distributed in a range [-0.15, 0.15]. For one simulation in which pictures are degraded only by noise, this range is [-0.3, 0.3]. Figure 17 shows results of degrading one of the training pictures with these methods. These are only examples of picture degradations, used for demonstrating the the system's functionality. **ANNIE** is capable of correcting other types of blur and noise as well.

Modular training

Every link array or group of arrays can be trained separately as a module. Modular training has many advantages. One is ease of testing. In a simpler network, local

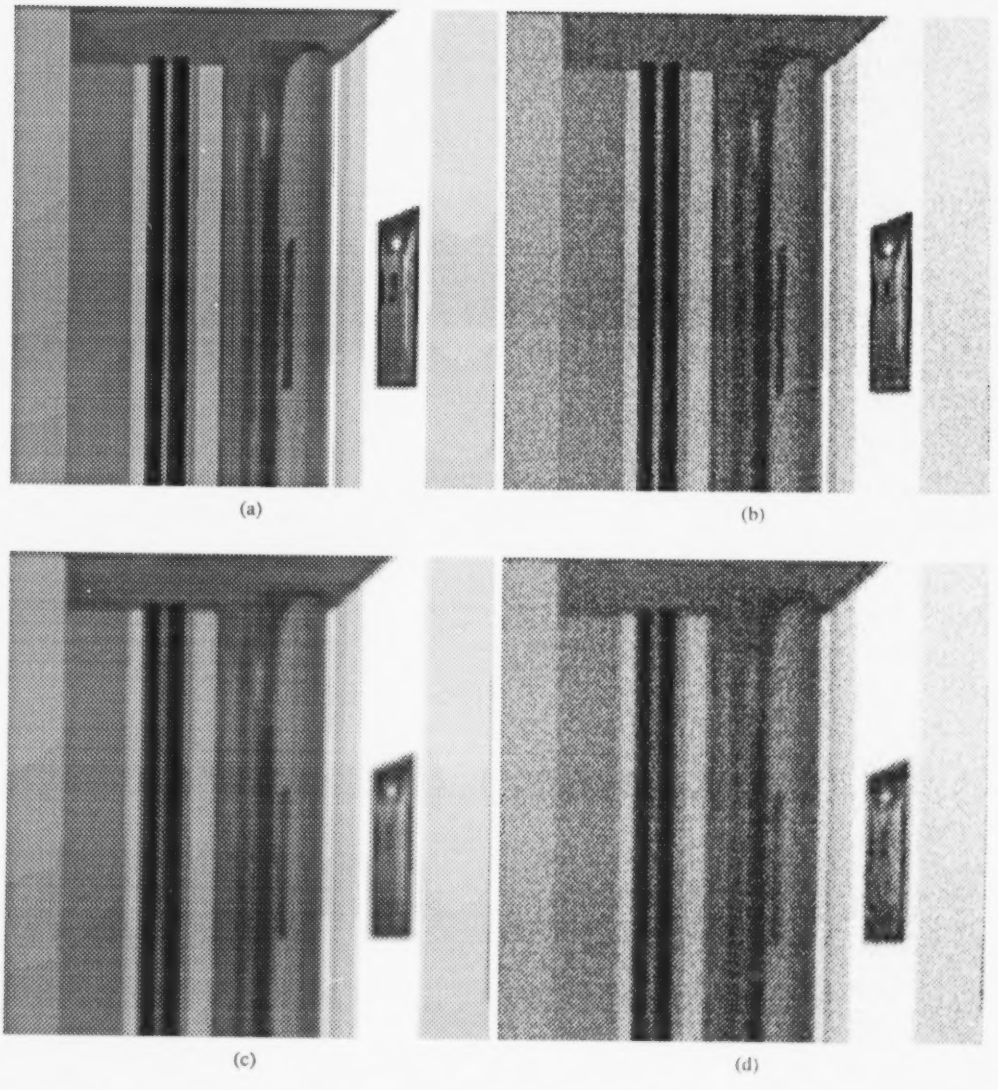


Figure 17: Degradations applied to one of the training patterns. (a) the original picture, (b) the same picture with noise, (c) with blur, (d) with both blur and noise.

minima are less probable and we can obtain estimations of the performance that can be reached. If the network in its full complexity does not show a better performance, it is because it has reached a local minimum or because of a programming error. For a simpler network, we can also estimate more easily what the weights should be, and what is the cause for not reaching that state. Another advantage of modular training is increased learning speed, as suggested by Hinton [14]. Learning times increase faster than linearly with the size of the networks. Therefore, for smaller modules training is faster than for the entire network. Eventually, training the entire network uses an initial state based on results from training each module separately. Training each module also can use results from training other modules.

Testing and debugging

Testing and debugging was an essential stage in developing **ANNIE** and it used several procedures. One of these procedures was visualizing the weights of links at intermediate stages of training. Other procedures were based on visualizing the outputs of layers and the errors propagated to layers as pictures. Sometimes, more complex operations with these pictures were used, such as detecting where the error is increasing from one step to another.

Weights Initialization

All the weights that are kept fixed are initialized with the desired values. These are the weights of the links that implement the Mexican-Hat filters and the weights of the links between the edge identification units and their duplicate zero-crossing units. The weight at the center of the input-output averaging weight array is initialized as equal to the unit. All the other weights are initialized as null. This initialization determines the system to create an initial output pattern identical to the input pattern. The initial error of the system is determined by the difference between the degraded picture and the reference picture. It is better to start the error's minimization from this value because it is much lower than the one obtained when all weights are null, and because it is hard to estimate a set of weights with

which error is lower.

Identifying Convergence and Stopping the Training Process

Experiments with training **ANNIE** point out a general behavior: during an initial stage, the mean square error decreases rapidly, until it is close to the minimal value that can be achieved; in a second stage, the mean square error decreases slowly and training after this point has low efficiency; eventually, in a third stage, the mean square error is practically oscillating slowly around a certain value, increasing slightly for a few cycles and then decreasing again for an approximately equal number of cycles. During the last stage, the weight changes are very small compared to the weights; in absolute value, the changes are about two size orders (or more) smaller than the weights themselves. It can be considered that the value around which the mean square error is oscillating during the third stage is the minimal value of convergence. However, it is not necessary to train the neural network to this point. Even during the second stage, another point is attained when the performance on the test pattern is beginning to decrease. Training was stopped after this point was attained; training more would be inefficient for the computation time and can cause overtraining of the neural network [1].

5.2 Simulation Results

This section presents only the results obtained when using **ANNIE** for noise removal and edge sharpening. The Appendices present the results obtained for other cases and some results on digitized real pictures. In discussing the results of the simulations, an important role is held by visualizing weights and by visualizing where the error is highest in the training and the test patterns. The weights determined through training are different when different types of image degradations are used. For each case, weights are visualized as white squares if positive and as black

[1] Overtraining a neural network means that the performance is improved for the training set of patterns, but generality is lost and performance on other patterns is decreasing.

Pattern	Initial Error	Final Error
Training pattern 1	.114	.0384
Training pattern 2	.0942	.0313
Training pattern 3	.115	.0358
Training pattern 4	.0941	.0295
Training pattern 5	.114	.0337
Training pattern 6	.0947	.0301
Training pattern 7	.114	.0323
Training pattern 8	.099	.0371
Training pattern 9	.123	.0624
Test pattern	.112	.0511

Table 5.2: The mean square error for noise removal and edge sharpening.

squares if negative. The sizes of the squares are proportional to the absolute values of the weights. Considering that the weights vary in a range $[-w, w]$, a maximum size white square represents a weight equal to w and a maximum size black square represents a weight equal to $-w$. Where two weight arrays are the transpose of each other (see Section 4.4) only one of them is represented. Error pictures show the differences between the actual outputs and the desired ones as lighter pixels where the error is positive and as darker pixels where the error is negative. The pixel intensities are determined by the absolute values of the error. Assuming that the error varies in a range $[-i, i]$, a white pixel represents an error i and a black pixel represents an error $-i$.

Table 5.2 presents the results from training to remove noise generated randomly in the range [-0.15, 0.15], and to sharpen edges at the same time. Figure 18 shows the output picture in response to the test pattern and compares it to the pattern used as input and to the pattern used as reference. Figure 19 illustrates the weights for different link arrays.

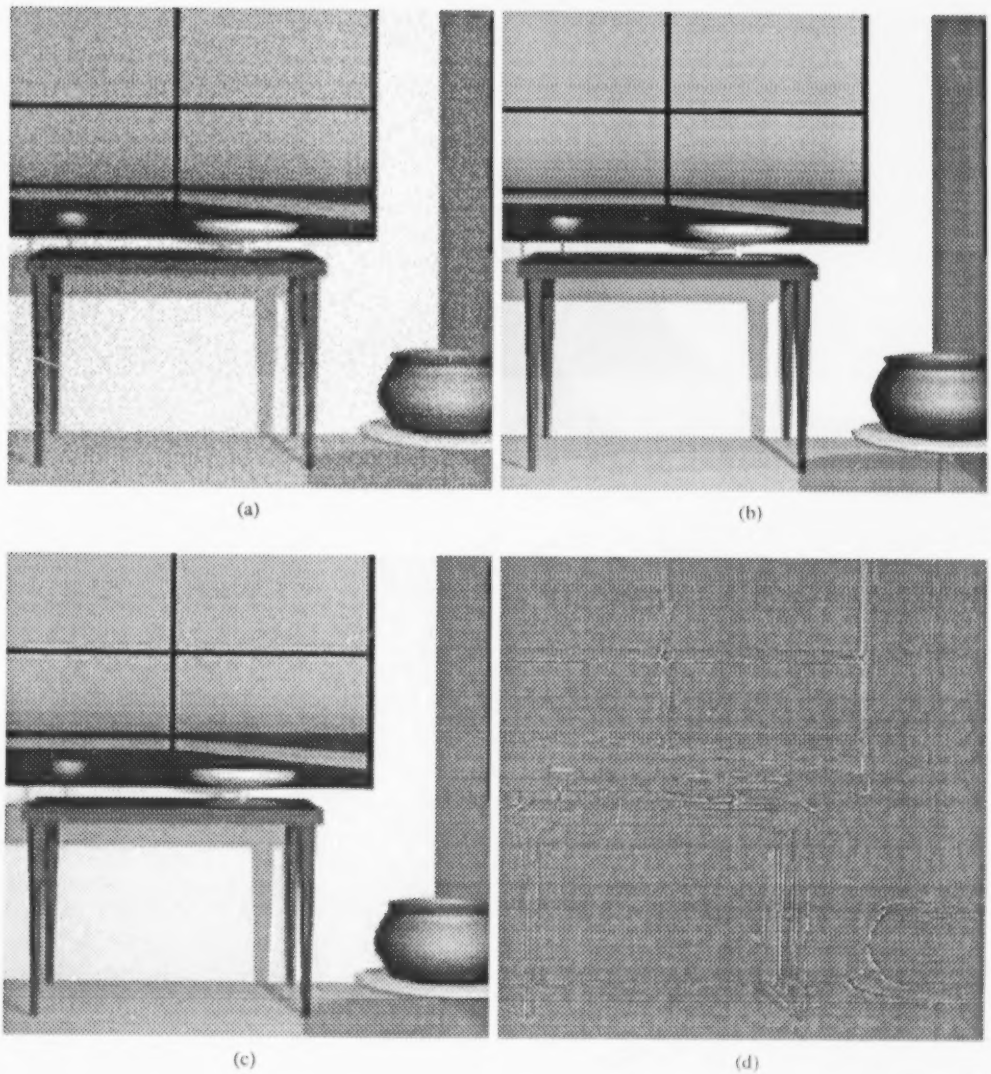


Figure 18: The output in response to the test pattern, compared to the input and to the reference, after training for noise removal and edge sharpening. (a) the input picture (clipped to the size of the reference and the output pictures), (b) the reference picture, (c) the output picture, (d) the difference between the output and the reference.

Figure 19: Weights obtained by training for noise removal and edge sharpening: the weights of the link arrays (EOA and EOB in Figure 7) between the output layer and the vertical edge identification layers of (a) Channel A (EOA), (b) Channel B (EOB). The maximum size squares represent weights that are equal to .0359 in absolute value.

Figure 19[Cont.]: Weights obtained by training for noise removal and edge sharpening: the weights of the link arrays (ICA and CCBA in Figure 7) between Channel A's vertical edge identification layers and (c) Channel A's vertical zero-crossing layers (ICA), (d) Channel A's horizontal zero-crossing layers (ICA), (e) Channel B's vertical zero-crossing layers (CCBA), (f) Channel B's horizontal zero-crossing layers (CCBA). The maximum size squares represent weights that are equal to .278 in absolute value.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 19[Cont.]: Weights obtained by training for noise removal and edge sharpening: the weights of the link arrays (ICB in Figure 7) between Channel B's vertical edge identification layers and (g) Channel B's vertical zero-crossing layers (the two weights symbolized by X have the value .848), (h) Channel B's horizontal zero-crossing layers; the weights of the link arrays (CCAB in Figure 7) between Channel B's vertical zero-crossing layers and (i) Channel A's vertical zero-crossing layers, (j) Channel A's horizontal zero-crossing layers. The maximum size squares represent weights that are equal to .447 in absolute value.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 19[Cont.]: Weights obtained by training for noise removal and edge sharpening: the weights of the input – output links (IO in Figure 7). The maximum size squares represent weights that are equal to .0405 in absolute value.

Simulation	Initial Error	Final Error
Edge sharpening in the presence of noise	.0705	.0437
Noise removal ([-0.15, 0.15])	.0865	.0352
Noise removal ([-0.3, 0.3])	.173	.0501
Noise removal and edge sharpening	.112	.0511
Edge sharpening	.0705	.0355

Table 5.3: The summary of the results for the test pattern.

5.3 Comparison of Results

Table 5.3 summarizes the results for the test pattern in all simulation cases. These results suggest that **ANNIE** performs best for noise removal. However, it should be pointed out that the test pattern has some sharp edges that are very close to each other and it is around those edges that error is highest. In such cases, blur eliminates a large amount of information that cannot be recuperated. Pictures with fewer small details would show better performance in removing blur.

How **ANNIE** adapts itself to different types of picture degradations becomes evident from comparing the weights of the link arrays for different simulations. For edge sharpening without any noise, the network relies almost completely on Channel A. This is proved by the fact that the weights of the links connecting the output layer to the edge identification layers are much larger in Channel A than in Channel

B. In the simulations with noise in the range [-0.15, 0.15], both channels are almost equally reliable. When noise is in the larger range [-0.3, 0.3], the network relies more on Channel B and therefore the corresponding weights of Channel B are larger. This variation is due to the fact that Channel A is more accurate in detecting details but is more sensitive to noise. Edge detectors from different channels having the same position and orientation inhibit each other and the link between them has a negative weight. This explains why when both channels are almost equally reliable, edges are detected by only one channel. This effect is evident in Figure 14. When no noise removal is performed, the input – output links learn to convey only the value of the input pixel to the corresponding output pixel and averaging of surrounding pixels is almost negligible. This averaging is used only for noise removal, and it is interesting to note that the shape of the averaging filter is flatter with higher noise. This means that the operation of averaging is optimized for different levels of noise.

Chapter 6

Conclusions

ANNIE represents a neural network method to remove noise and to sharpen edges in digital pictures. It is difficult to compare it to other methods because there is no numerical measure for performance in the two operations of image enhancement. However, an advantage is that, when the degradation can be reproduced on a set of training patterns, a neural network is capable of adapting to optimize the enhancement. On the other hand, the long duration of the training process is a disadvantage. Even used normally, after training, **ANNIE** is much slower than most other noise removal and edge enhancement operations, because it involves complex operations like edge detection and edge relaxation.

Because noise removal based on averaging and edge sharpening based on edge detection are very general image enhancement methods, **ANNIE** can be used with similar results for any type of noise and blur. In some cases, other methods may perform better. Appendix D presents a visual comparison with median filtering, a noise removal method that preserves well edges [37]. For the specific type of noise used, **ANNIE** performs better. For some digitized real pictures, good results were obtained in removing sampling noise. For a thorough performance assessment, further experiments using more types of noise and blur are necessary to compare **ANNIE** with other noise removal, edge enhancement, and edge detection methods.

ANNIE represents also a neural network method for edge detection and is a

natural physical implementation for the study of Marr's theory of edge detection. Features adopted from this theory are the multiple channels and zero-crossing detection. However, the channel interaction is based on lateral inhibition, and not on the detection of features like lines, blobs, and terminations, as proposed by Marr. Experiments can show the advantages of using multiple channels, channel interaction and edge relaxation. **ANNIE** can simulate the action of a single channel by training only the links of that channel and keeping the weights in the other channel equal to 0. Inter- and intra-channel interaction can be disabled by keeping the weights of all the corresponding links equal to 0.

The implementation presented in this thesis is meant mainly to illustrate the image enhancement and edge detection methods that it uses, and to exemplify how a neural network is capable of learning features like filter shapes and edge relaxation. Its performance is not ideal and it is open to many improvements. The present implementation shows good results in removing noise. However, the averaging operation introduces an additional blur. Long, straight, and isolated edges are reconstructed almost completely, but fine details, curved edges and narrow bars (lines) are enhanced poorly. There are several ways to implement a higher performance neural network for both image enhancement and edge detection:

1. Two edge detectors could be implemented for each orientation, detecting separately edges of opposite directions of contrast. Adding this information that is not used in **ANNIE**, can make edge relaxation more accurate. Indeed, two edge detectors with the same orientation and lined up on their orientation should reinforce each other if they have also the same contrast direction, but should inhibit each other if they have opposite contrast directions.

2. More stages can be used in implementing the recurrent interaction between edge detectors. As false edges are elliminated in successive stages, the remaining edges contain more pertaining information and the interaction between edge detectors is more accurate.

3. Edge detectors with more than two orientations can be used. Two more

diagonal orientations are easy to implement. This allows more accuracy in the edge detection and in edge sharpening.

4. More than two edge detection channels can be used. This would enable the neural network to detect edges even in pictures with much stronger noise and/or blur. A channel with a very small space constant would be especially useful, because it will improve edge detection in regions with small details, where the performance of **ANNIE** is poorest.

5. A more extensive set of training patterns is necessary. More training is necessary for curved edges and inclined edges, both isolated and close to each other. Real pictures should be added to the artificial ones.

6. The training process can be extended to all the links (see Figure 7). Adjusting the weights of the MZA and MZB links would allow a weighted averaging of the Mexican-hat filter outputs instead of averaging according to Equations 3.7-3.10; the effect would be a better accuracy in measuring the intensity of zero-crossings and a better separation between noise edges and real edges. Adjusting the weights of the IMA and IMB links would allow the neural network to choose the optimal space constants for the two channels.

These changes have the disadvantage that they would make the neural network more complex and slower to train.

The most important problems that emerged in developing **ANNIE** are related to training speed and testability. Training the neural network needs thousands of CPU hours on a SUN SPARCstation. This heavy running time is caused by the complexity of the network (5771 links, of which 5217 have learned weights), and to the size of the training patterns which have to include a high number of features. Modular training proved itself to be very efficient in reducing the training time, especially in terms of CPU time (as opposed to number of training cycles). Testing and debugging were difficult stages in developing the neural network because we cannot anticipate results against which the actual results can be compared. The

experience accumulated while developing **ANNIE** suggests that a set of testing procedures should be created in a very early stage. This set should include tests with increasing degrees of complexity; results of complex tests are difficult to interpret and therefore, more simple tests should be performed first to eliminate most of the errors in implementation.

Pictures transmitted by satellites are an example of a possible application. Such pictures are usually affected by noise. A set of known training patterns can be transmitted periodically from the satellite and they can be used to retrain the neural network. The knowledge acquired this way can be used to enhance the pictures that are transmitted between two consecutive transmissions of the training patterns.

A similar system can be implemented to perform only edge detection. In this case the input - output link array has to be formed of only one link and the link arrays between the edge identification layers and the output layer can be smaller, probably 4×4. The neural network would be trained to blur pictures instead of deblurring them. This can be done by using the original patterns as inputs and the blurred patterns as reference at the output. Without the possibility of averaging a neighborhood of pixels, the neural network should learn to detect edges and to add a blurring correction around them.

Bibliography

[1] A. Rosenfeld and A. C. Kak (1976). *Digital Picture Processing.* Academic Press, New York, NY.

[2] W. K. Pratt (1978). *Digital Image Processing.* John Wiley & Sons.

[3] L. S. Davis (1975). A Survey of Edge Detection Techniques. *Computer Graphics and Image Processing*, **4**, 248–270.

[4] T. N. Cornsweet (1970). *Visual Perception.* Academic Press, New York, NY.

[5] J. E. Dowling (1987). *The Retina: An Approachable Part of the Brain.* Belknap Press of Harvard University, Cambridge, Mass.

[6] E. R. Kandel and J. H. Schwartz (1981). *Principles of Neural Science*, pages 213–248. Elsevier North Holland, New York, NY.
Based on Dowling's work, but more succinct and more accessible.

[7] D. H. Hubel and T. N. Wiesel (1962). Receptive Fields, Binocular Interaction, and Functional Architecture in the Cat's Visual Cortex. *Journal of Physiology*, **160**, 106–154.

[8] D. H. Hubel and T. N. Wiesel (1974). Uniformity of Monkey Striate Cortex: A Parallel Relationship Between Field Size, Scatter, and Magnification Factor. *Journal of Comparative Neurology*, **158**, 295–305.
Experiments related to different receptive-field sizes.

[9] O. Braddick, F. W. Campbell, and J. Atkinson (1978). Channels in Vision: Basic Aspects. In R. Held, H. W. Leibowitz, and H. Teuber (Eds.), *Handbook of Sensory Physiology*, volume 8, chapter 1, pages 3–38. Springer-Verlag, Berlin-Heidelberg-New York.

[10] F. W. Campbell and J. G. Robson (1968). Application of Fourier Analysis to the Visibility of Gratings. *Journal of Physiology*, **197**, 551–566.
Psychophysical evidence for independent mechanisms selectively sensitive to limited ranges of spatial frequency.

[11] S. W. Kuffler, J. G. Nicholls, and A. R. Martin (1984). *From Neuron to Brain*, chapter 1, pages 34–47. Sinauer, Sunderland, Mass.

[12] R. Hecht-Nielsen (1988). Neurocomputing: Picking the Human Brain. *IEEE Spectrum*, **25**, 36–41.

[13] D. E. Rumelhart and J. L. McClelland (1986). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. MIT Press, Cambridge, Mass.

[14] G. E. Hinton (1989). Connectionist Learning Procedures. *Artificial Intelligence*, **40**, 185–234.

[15] R. P. Lippmann (1987). An Introduction to Computing with Neural Nets. *IEEE ASSP Magazine*, **4**, 4–22.

[16] R. Hecht-Nielsen (1989). Theory of the Backpropagation Neural Network. In *Proceedings of the 1989 IJCNN* (pp. I:593–605). IEEE.
One of the best overviews of backpropagation. Well written with clear formal definition of the backpropagation architecture. The approach based on the error surface allows a better understanding of the backpropagation mechanisms. Hecht-Nielsen is against variants that yet can increase the speed of learning.

[17] R. M. Shapley and D. J. Tolhurst (1973). Edge Detectors in Human Vision. *Journal of Physiology*, **229**, 165–183.

[18] D. C. Marr (1982). *Vision.* W. H. Freeman, San Francisco, CA.

[19] D. C. Marr and E. Hildreth (1980). Theory of Edge Detection. *Proceedings of the Royal Society of London*, **B 207**, 187–217.

[20] D. C. Marr, T. Poggio, and S. Ullman (1979). Bandpass Channels, Zero-Crossings, and Early Visual Information Processing. *Journal of Optical Society of America*, **69**, 914–916.

[21] F. Glazer (1983). Multilevel Relaxation in Low Level Computer Vision. In A. Rosenfeld (Ed.), *Multiresolution Image Processing.* Springer-Verlag, Berlin-Heidelberg-New York.

[22] F. Glazer, G. Reynolds, and P. Anandan (1983). Scene Matching by Hierarchical Correlation. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference.*

[23] A. Rosenfeld (Ed.). *Multiresolution Image Processing.* Springer-Verlag, Berlin-Heidelberg-New York.

[24] G.A. Carpenter, S. Grossberg, and C. Mehanian (1989). Invariant Recognition of Cluttered Scenes by a Self-Organizing ART Architecture: CORT-X Boundary Segmentation. *Neural Networks*, **2**, 169–181.

[25] M. B. Sachs, J. Nachmias, and J. G. Robson (1971). Spatial-Frequency Channels in Human Vision. *Journal of the Optical Society of America*, **61**, 1176–1186.

[26] A. Rosenfeld, R. A. Hummel, and S. W. Zucker (1976). Scene Labeling by Relaxation Operations. *IEEE Transactions on Systems, Man and Cybernetics*, **6**, 420–433.

[27] D. H. Ballard and C. Brown (1982). *Computer Vision.* Prentice-Hall, Englewood Cliffs, NJ.

[28] G. E. Hinton (1977). *Relaxation and Its Role in Vision.* PhD thesis, University of Edinburgh.

[29] S. Grossberg and E. Mingolla (1985). Neural Dynamics of Form Perception: Boundary Completion, Illusory Figures, and Neon Color Spreading. *Psychological Review*, **92**, 173–211.
Model for edge detection using relaxation, but it is algorithmically programmed in a large measure.

[30] W. Bair and C. Koch (1991). An Analog VLSI Chip for Finding Edges from Zero-crossings. In D.S. Touretzky and R. Lippman (Eds.), *Advances in Neural Information Processing Systems 3*, M. Kaufmann, San Mateo, CA.

[31] N.H. Goddard, K.J. Lynne, T. Mintz, and L. Bukys (1989). Rochester Connectionist Simulator. Technical Report 233, The University of Rochester, Computer Science Department, Rochester, NY.

[32] S.E. Fahlman (1989). Faster-Learning Variations on Back-Propagation: An Empirical Study. In D.S. Touretzky, G. Hinton, and T. Sejnowski (Eds.), *Proceedings of the 1988 Connectionist Models Summer School*, M. Kaufmann, San Mateo, CA.

[33] R.A. Jacobs (1988). Increased Rates of Convergence Through Learning Rate Adaptation. *Neural Networks*, **1**, 295–307.

[34] D.B. Parker (1987). Optimal Algorithms for Adaptive Networks: Second Order Back Propagation, Second Order Direct Propagation, and Second Order Hebbian Learning. In *Proceedings of the IEEE International Conference on Neural Networks*. San Diego, CA.

[35] R.L. Watrous (1987). Learning Algorithms for Connectionist Networks: Applied Gradient Methods for Non-Linear Optimization. In *Proceedings of the IEEE International Conference on Neural Networks*. San Diego, CA.

[36] S. Becker and Y. LeCun (1989). The Feasibility of Applying Numerical Optimization Techniques to Backpropagation. In D.S. Touretzky, G. Hinton, and T. Sejnowski (Eds.), *Proceedings of the 1988 Connectionist Models Summer School*, M. Kaufmann. San Mateo, CA.

[37] R. C. Gonzales and P. Wintz (1987). *Digital Image Processing*, page 162. Addison Wesley.

Appendix A

Edge Sharpening in the Presence of Noise

Two simulations were used to determine the optimal sizes for the link arrays between hidden layers. In the first simulation, the links going into the zero-crossing and edge identification layers of the Channel A form arrays of 43×43 links, and the links going into the zero-crossing and edge identification layers of the Channel B form arrays of 15×15 links. These arrays cover all possible interferences from other edges. After training, only some weights in the center of the arrays are significant and only the corresponding links were used in the final version of **ANNIE**, as represented in Table 3.1. Because this simulation was used only for edge sharpening, the input–output link array is limited to only one link having the weight equal to 1. Also, because added blur is placed only adjacent to edges, the edge identification–output link arrays are 4×4. The advantage in using limited link arrays is reduced computation time for the simulations. Noise is generated randomly in the range [-0.15, 0.15].

Table A.4 presents the mean square error before and after training the larger structure, measured for the training patterns and for the test pattern. By reducing the intra-channel and inter-channel link arrays to the sizes from Table 3.1, the loss in accuracy is very small. Table A.5 presents the results from this simulation.

Pattern	Initial Error	Final Error
Training pattern 1	.0682	.0277
Training pattern 2	.0342	.0194
Training pattern 3	.069	.0237
Training pattern 4	.0343	.0162
Training pattern 5	.0686	.0219
Training pattern 6	.0346	.0171
Training pattern 7	.0692	.0199
Training pattern 8	.0455	.0207
Training pattern 9	.0819	.047
Test pattern	.0705	.0462

Table A.4: The mean square error for edge sharpening in the presence of noise (simulation with an experimental, larger network structure).

The simulation time for one training cycle with this structure is about four times faster than with the experimental, larger structure mentioned above. More than that, the simulation with the larger structure takes more training cycles because backpropagation scales poorly.

Finally, Table A.6 presents the results from training the normal structure of **ANNIE**, having all the link array sizes as in Table 3.1. Figure 20 shows the output picture in response to the test pattern and compares it to the pattern used as input and to the pattern used as reference. Figure 21 visualizes the weights for different link arrays. The weights for the input–output link array are not presented; the weight in the center is equal to .955 and the other weights are comparatively negligible.

Pattern	Final Error
Training pattern 1	.0273
Training pattern 2	.0183
Training pattern 3	.0241
Training pattern 4	.0171
Training pattern 5	.0231
Training pattern 6	.016
Training pattern 7	.0202
Training pattern 8	.0215
Training pattern 9	.0505
Test pattern	.0435

Table A.5: The mean square error for edge sharpening in the presence of noise (simulation with a reduced network structure).

Pattern	Final Error
Training pattern 1	.0305
Training pattern 2	.0194
Training pattern 3	.0235
Training pattern 4	.0169
Training pattern 5	.0218
Training pattern 6	.0186
Training pattern 7	.0205
Training pattern 8	.0216
Training pattern 9	.049
Test pattern	.0437

Table A.6: The mean square error for edge sharpening in the presence of noise.

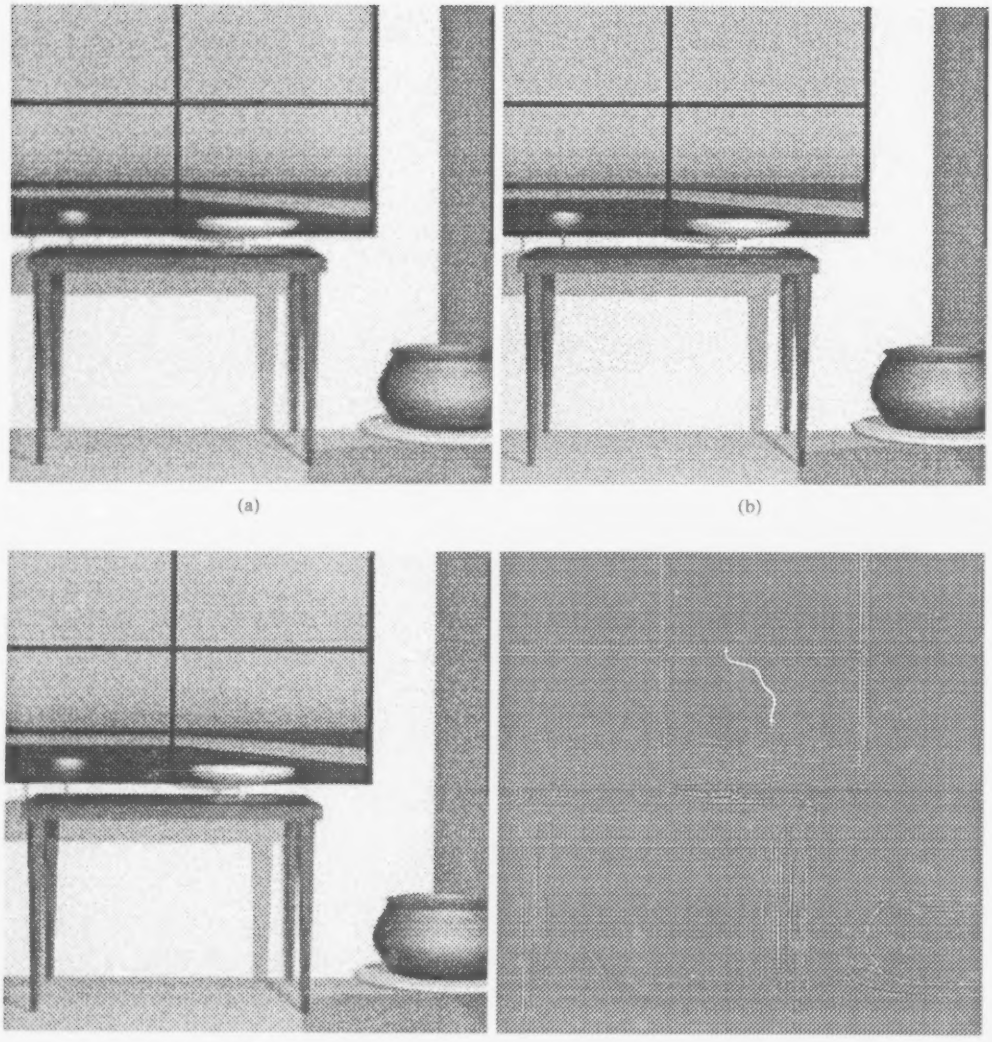


Figure 20: The output in response to the test pattern, compared to the input and to the reference, after training for edge sharpening in the presence of noise. (a) the input picture (clipped to the size of the reference and the output pictures), (b) the reference picture, (c) the output picture, (d) the difference between the output and the reference.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 21: Weights obtained by training for edge sharpening in the presence of noise: the weights of the link arrays (EOA and EOB in Figure 7) between the output layer and the vertical edge identification layers of (a) Channel A (EOA), (b) Channel B (EOB). The maximum size squares represent weights that are equal to .0416 in absolute value.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 21[Cont.]: Weights obtained by training for edge sharpening in the presence of noise: the weights of the link arrays (ICA and CCBA in Figure 7) between Channel A's vertical edge identification layers and (c) Channel A's vertical zero-crossing layers (ICA), (d) Channel A's horizontal zero-crossing layers (ICA), (e) Channel B's vertical zero-crossing layers (CCBA), (f) Channel B's horizontal zero-crossing layers (CCBA). The maximum size squares represent weights that are equal to .304 in absolute value.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 21[Cont.]: Weights obtained by training for edge sharpening in the presence of noise: the weights of the link arrays (ICB in Figure 7) between Channel B's vertical edge identification layers and (g) Channel B's vertical zero-crossing layers (the two weights symbolized by X have the value .872), (h) Channel B's horizontal zero-crossing layers; the weights of the link arrays (CCAB in Figure 7) between Channel B's vertical zero-crossing layers and (i) Channel A's vertical zero-crossing layers, (j) Channel A's horizontal zero-crossing layers. The maximum size squares represent weights that are equal to .294 in absolute value.

Appendix B

Noise Removal

Table B.7 presents the results from training to remove noise generated randomly in the range [-0.15, 0.15]. Figure 22 shows the output picture in response to the test pattern and compares it to the pattern used as input and to the pattern used as reference. Figure 23 illustrates the weights for different link arrays.

Table B.8 presents the results from training to remove noise generated randomly in the range [-0.3, 0.3]. Figure 24 shows the output picture in response to the test pattern and compares it to the pattern used as input and to the pattern used as reference. Figure 25 illustrates the weights for different link arrays.

Pattern	Initial Error	Final Error
Training pattern 1	.086	.0286
Training pattern 2	.0863	.0226
Training pattern 3	.0868	.0267
Training pattern 4	.087	.022
Training pattern 5	.0871	.0253
Training pattern 6	.0866	.0232
Training pattern 7	.0868	.0239
Training pattern 8	.086	.0311
Training pattern 9	.0861	.0451
Test pattern	.0865	.0352

Table B.7: The mean square error for noise removal with noise generated randomly in the range [-0.15, 0.15].

Pattern	Initial Error	Final Error
Training pattern 1	.173	.0411
Training pattern 2	.172	.0371
Training pattern 3	.174	.0418
Training pattern 4	.173	.0372
Training pattern 5	.175	.0402
Training pattern 6	.174	.0366
Training pattern 7	.173	.0383
Training pattern 8	.173	.045
Training pattern 9	.173	.0609
Test pattern	.173	.0501

Table B.8: The mean square error for noise removal with noise generated randomly in the range [-0.3, 0.3].

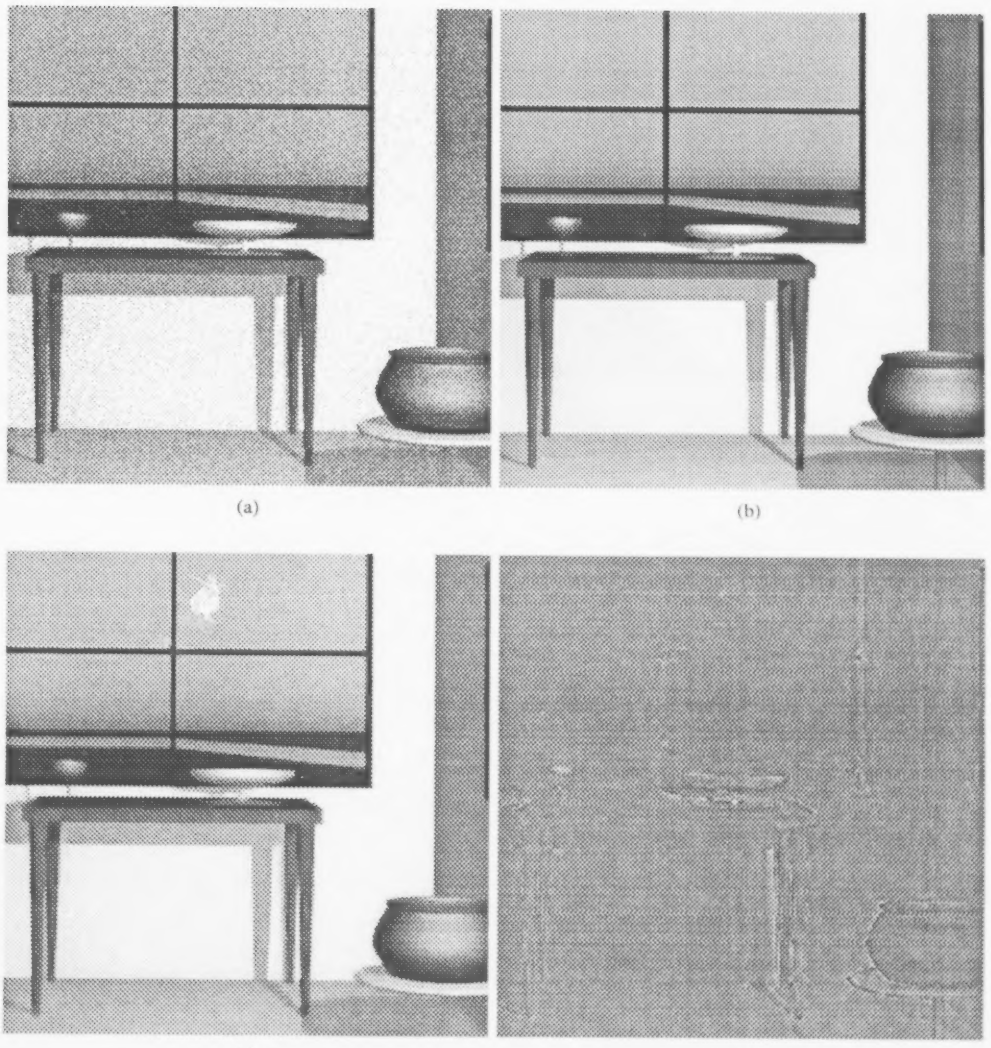


Figure 22: The output in response to the test pattern, compared to the input and to the reference, after training for noise removal with noise generated randomly in the range [-0.15, 0.15]. (a) the input picture (clipped to the size of the reference and the output pictures), (b) the reference picture, (c) the output picture, (d) the difference between the output and the reference.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 23: Weights obtained by training for removing noise generated randomly in the range [-0.15, 0.15]: the weights of the link arrays (EOA and EOB in Figure 7) between the output layer and the vertical edge identification layers of (a) Channel A (EOA), (b) Channel B (EOB). The maximum size squares represent weights that are equal to .0208 in absolute value.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 23[Cont.]: Weights obtained by training for removing noise generated randomly in the range [-0.15, 0.15]: the weights of the link arrays (ICA and CCBA in Figure 7) between Channel A's vertical edge identification layers and (c) Channel A's vertical zero-crossing layers (ICA), (d) Channel A's horizontal zero-crossing layers (ICA), (e) Channel B's vertical zero-crossing layers (CCBA), (f) Channel B's horizontal zero-crossing layers (CCBA). The maximum size squares represent weights that are equal to .424 in absolute value.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 23[Cont.]: Weights obtained by training for removing noise generated randomly in the range [-0.15, 0.15]: the weights of the link arrays (ICB in Figure 7) between Channel B's vertical edge identification layers and (g) Channel B's vertical zero-crossing layers, (h) Channel B's horizontal zero-crossing layers; the weights of the link arrays (CCAB in Figure 7) between Channel B's vertical zero-crossing layers and (i) Channel A's vertical zero-crossing layers, (j) Channel A's horizontal zero-crossing layers. The maximum size squares represent weights that are equal to .27 in absolute value.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 23[Cont.]: Weights obtained by training for removing noise generated randomly in the range [-0.15, 0.15]: the weights of the input–output links (IO in Figure 7). The maximum size squares represent weights that are equal to .0289 in absolute value.

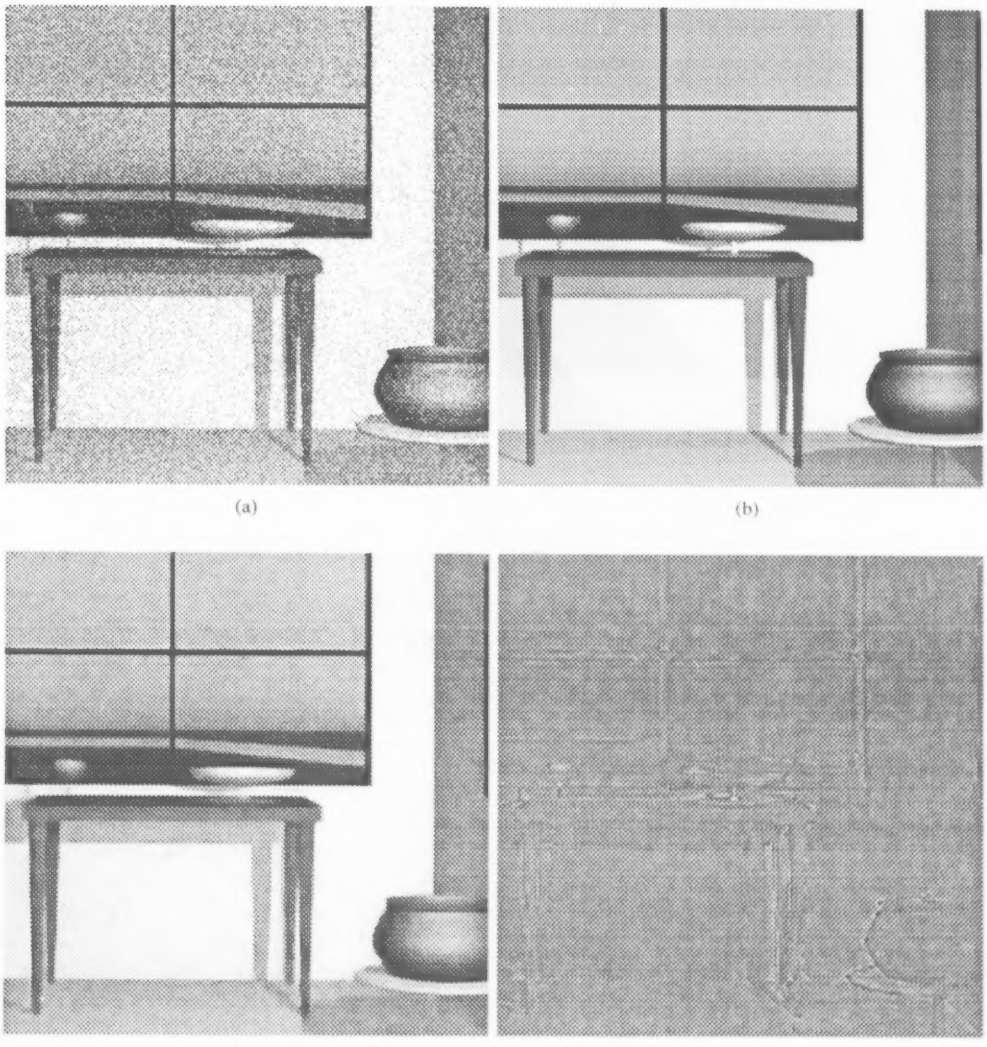


Figure 24: The output in response to the test pattern, compared to the input and to the reference, after training for noise removal with noise generated randomly in the range [-0.3, 0.3]. (a) the input picture (clipped to the size of the reference and the output pictures), (b) the reference picture, (c) the output picture, (d) the difference between the output and the reference.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 25: Weights obtained by training for removing noise generated randomly in the range [-0.3, 0.3]: the weights of the link arrays (EOA and EOB in Figure 7) between the output layer and the vertical edge identification layers of (a) Channel A (EOA), (b) Channel B (EOB). The maximum size squares represent weights that are equal to .0182 in absolute value.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 25[Cont.]: Weights obtained by training for removing noise generated randomly in the range [-0.3, 0.3]: the weights of the link arrays (ICA and CCBA in Figure 7) between Channel A's vertical edge identification layers and (c) Channel A's vertical zero-crossing layers (ICA), (d) Channel A's horizontal zero-crossing layers (ICA), (e) Channel B's vertical zero-crossing layers (CCBA), (f) Channel B's horizontal zero-crossing layers (CCBA). The maximum size squares represent weights that are equal to .282 in absolute value.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 25[Cont.]: Weights obtained by training for removing noise generated randomly in the range [-0.3, 0.3]: the weights of the link arrays (ICB in Figure 7) between Channel B's vertical edge identification layers and (g) Channel B's vertical zero-crossing layers, (h) Channel B's horizontal zero-crossing layers; the weights of the link arrays (CCAB in Figure 7) between Channel B's vertical zero-crossing layers and (i) Channel A's vertical zero-crossing layers, (j) Channel A's horizontal zero-crossing layers. The maximum size squares represent weights that are equal to .615 in absolute value.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 25[Cont.]: Weights obtained by training for removing noise generated randomly in the range [-0.3, 0.3]: the weights of the input–output links (IO in Figure 7. The maximum size squares represent weights that are equal to .0317 in absolute value.

Appendix C

Edge Sharpening

Table C.9 presents the results from training to sharpen edges. Figure 26 shows the output picture in response to the test pattern and compares it to the pattern used as input and to the pattern used as reference. Figure 27 illustrates the weights for different link arrays. The weights for the input–output link array are not represented; the weight in the center is equal to .958 and the other weights are comparatively negligible.

Pattern	Initial Error	Final Error
Training pattern 1	.0682	.0242
Training pattern 2	.0342	.0114
Training pattern 3	.069	.0117
Training pattern 4	.0343	.0103
Training pattern 5	.0686	.0134
Training pattern 6	.0346	.00837
Training pattern 7	.0692	.0101
Training pattern 8	.0455	.0111
Training pattern 9	.0819	.0444
Test pattern	.0705	.0355

Table C.9: The mean square error for edge sharpening.

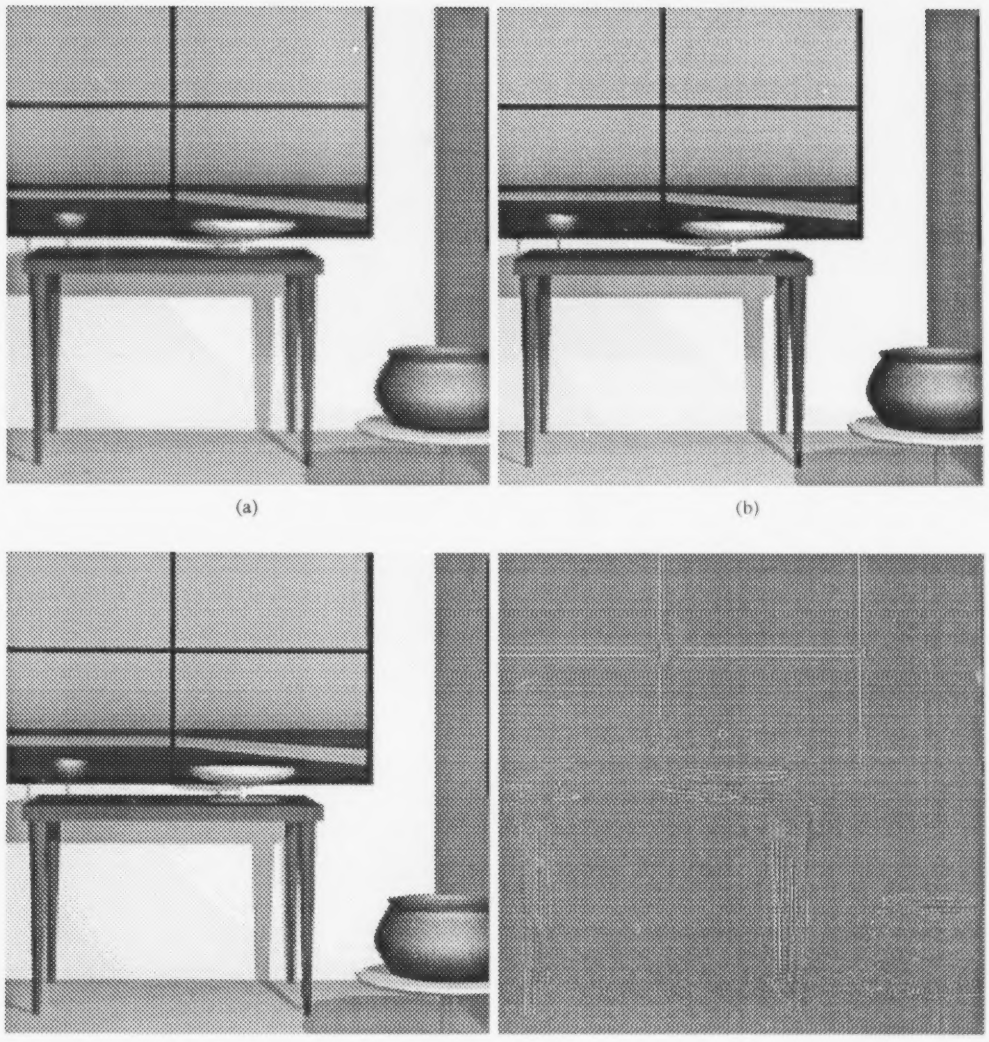


Figure 26: The output in response to the test pattern, compared to the input and to the reference, after training for edge sharpening. (a) the input picture (clipped to the size of the reference and the output pictures), (b) the reference picture, (c) the output picture, (d) the difference between the output and the reference.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 27: Weights obtained by training for edge sharpening: the weights of the link arrays (EOA and EOB in Figure 7) between the output layer and the vertical edge identification layers of (a) Channel A (EOA), (b) Channel B (EOB). The maximum size squares represent weights that are equal to .117 in absolute value.

FIGURE REMOVED BECAUSE OF
POOR REPRODUCTION QUALITY

Figure 27[Cont.]: Weights obtained by training for edge sharpening: the weights of the link arrays (ICA and CCBA in Figure 7) between Channel A's vertical edge identification layers and (c) Channel A's vertical zero-crossing layers (ICA), (d) Channel A's horizontal zero-crossing layers (ICA), (e) Channel B's vertical zero-crossing layers (CCBA), (f) Channel B's horizontal zero-crossing layers (CCBA). The maximum size squares represent weights that are equal to .228 in absolute value.

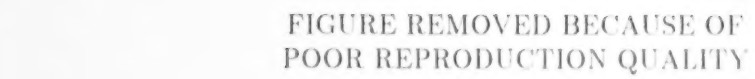


Figure 27[Cont.]: Weights obtained by training for edge sharpening: the weights of the link arrays (ICB in Figure 7) between Channel B's vertical edge identification layers and (g) Channel B's vertical zero-crossing layers (the two weights symbolized by X have the value .873), (h) Channel B's horizontal zero-crossing layers; the weights of the link arrays (CCAB in Figure 7) between Channel B's vertical zero-crossing layers and (i) Channel A's vertical zero-crossing layers, (j) Channel A's horizontal zero-crossing layers. The maximum size squares represent weights that are equal to .664 in absolute value.

Appendix D

Using ANNIE on Real Pictures

Training was made only with artificially generated pictures. Digitized real pictures have more fine details, more curved edges, and less sharp edges, features that are weakly represented in the training set. Tests show that the knowledge acquired through learning can be applied only with lower performance to digitized real pictures. For better performance, a more extensive set of training patterns is necessary, including real pictures with the features mentioned above.

Figure 28 shows the results achieved on two digitized real pictures. Enhancement is poorer in the regions with fine details. It is important to stress however, that noise and blur can make these details hard to recover by any other method. Some curved edges tend to be enhanced as straight vertical or horizontal edges because these are the two orientations detected by **ANNIE** and because training did not include a sufficient number of curved edges. As in the artificial pictures, noise is almost completely removed, and long, straight, and isolated edges are well sharpened.

Figure 29 compares **ANNIE** to median filtering, a method widely used for noise removal [37]. The picture used as input in all cases has random noise in the range [-0.15, 0.15], as described in Section 5.1. At least for this type of noise, **ANNIE** gives a better result. However, in other cases (for example, impulse noise – where only a certain percentage of the pixels are degraded), median filtering may perform better.

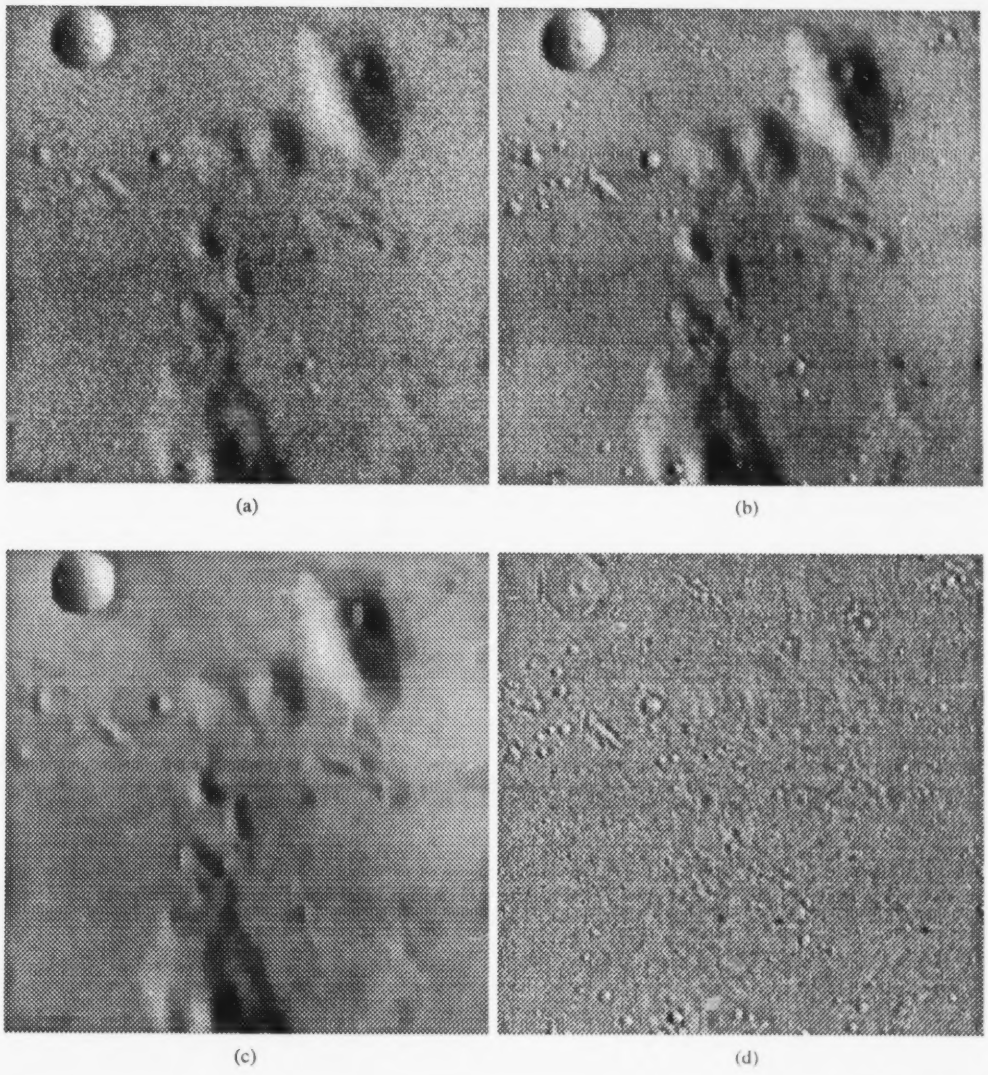


Figure 28: Results obtained on real pictures. (a) the input picture (clipped to the size of the reference and the output pictures), (b) the reference picture, (c) the output picture, (d) the difference between the output and the reference.

(e) (f)

(g) (h)

Figure 28[Cont.]: Results obtained on real pictures. (e) the input picture (clipped to the size of the reference and the output pictures), (f) the reference picture, (g) the output picture, (h) the difference between the output and the reference.

Figure 29: Comparison between **ANNIE** and median filtering. (a) the input picture; the outputs of (b) **ANNIE**, (c) 3×3 median filter, (d) 5×5 median filter.